DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Desc
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exa** • `Art Will Make You H` • `First Grad`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fo enumerated v • `Grades P` • `Grade` • `Grade` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fr following enumerated list of v • `Applied Lea` • `Care & H` • `Health & S` • `History & C` • `Literacy & Lan` • `Math & Sc` • `Music & The` • `Special` • `W` **Exar** • `Music & The` • `Literacy & Language, Math & Sc`

Feature	Desc
`school_state`	State where school is located (Two-letter U.S. posta (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c **Exampl**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the p **Exan** • Lit • Literature & Writing, Social Sci
`project_resource_summary`	An explanation of the resources needed for the project. **Exa** • My students need hands on literacy materials to ma sensory needs!<
`project_essay_1`	First application
`project_essay_2`	Second application
`project_essay_3`	Third application
`project_essay_4`	Fourth application
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** 2016-0 12:43:5
`teacher_id`	A unique identifier for the teacher of the proposed project. **Ex** bdf8baa8fedef6bfeec7ae4ff1c
`teacher_prefix`	Teacher's title. One of the following enumerated v • • • • • • Tea
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same te **Exam**

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

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.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

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/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 plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 's
chool_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']
```

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

In [6]:

```
project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)

project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data["project_grade_category"] = project_grade_category
project_data.head(5)
```

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

1.2 preprocessing of `project_subject_categories`

In [7]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of project_subject_subcategories

In [8]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

Clean Titles (Text preprocessing)

In [9]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [10]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [11]:

```
clean_titles = []

for titles in tqdm(project_data["project_title"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    clean_titles.append(title.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 35375.52it/s]
```

In [12]:

```
project_data["clean_titles"] = clean_titles
```

In [13]:

```
project_data.drop(['project_title'], axis=1, inplace=True)
```

1.3 Text preprocessing

In [14]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [15]:

```
project_data.head(2)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

Clean Essays (Text preprocessing)

In [16]:

```
clean_essay = []

for ess in tqdm(project_data["essay"]):
    ess = decontracted(ess)
    ess = ess.replace('\\r', ' ')
    ess = ess.replace('\\"', ' ')
    ess = ess.replace('\\n', ' ')
    ess = re.sub('[^A-Za-z0-9]+', ' ', ess)
    ess = ' '.join(f for f in ess.split() if f not in stopwords)
    clean_essay.append(ess.lower().strip())
```

```
100%|██████████| 109248/109248 [01:10<00:00, 1549.33it/s]
```

In [17]:

```
project_data['clean_essays'] = clean_essay
```

In [18]:

```
project_data.drop(['essay'], axis=1, inplace=True)
```

In [19]:

```
project_data['total_txt'] = project_data['clean_titles'] + ' ' + project_data['clean_essays
```

In [20]:

```
project_data.head(5)
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

In [21]:

```
glove_vec = open('glove_vectors', 'rb')
gv = pickle.load(glove_vec)
```

Type *Markdown* and LaTeX: α^2

In [22]:

```
# train test split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['project_is_
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [23]:

```
print("Shape of the Train dataset: ", X_train.shape,y_train.shape)
print("Shape of the Test dataset: ", X_test.shape,y_test.shape)
print("Shape of the cv dataset:", X_cv.shape,y_cv.shape)
```

```
Shape of the Train dataset:  (49041, 19) (49041,)
Shape of the Test dataset:  (36052, 19) (36052,)
Shape of the cv dataset: (24155, 19) (24155,)
```

In [24]:

```
#converting class labels to categorical variables
from keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_cv = to_categorical(y_cv)
```

```
Using TensorFlow backend.
```

In [25]:

```
y_train.shape
```

Out[25]:

```
(49041, 2)
```

In [26]:

```
clean_text=[]
clean_text = project_data["total_txt"]
clean_text
```

Out[26]:

```
55660     engineering steam primary classroom i fortunat...
76127     sensory tools focus imagine 8 9 years old you ...
51140     mobile learning mobile listening center having...
473       flexible seating flexible learning i recently ...
41558     going deep the art inner thinking my students ...
29891     breakout box ignite engagement it end school y...
81565     flexible seating an environment help kids lear...
79026     21st century learning multimedia it not enough...
23374     ipad learners never society rapidly changed te...
86551     dash dot robotic duo needed do remember first ...
49228     a flexible classroom flexible minds my student...
72638     make powerful movies media cinematography extr...
7176      robots taking 2nd grade computer coding roboti...
70898     time kids to learn about science i teach 4th g...
102755    stem books games kits explore world in classro...
72593     getting plugged learning a typical day classro...
35006     help us travel world virtually we love technol...
100222    engaging readers with technology teachers love...
5145      books power powerful book clubs do remember bo...
48237     choice novels freshman students needed my stud...
64637     pre k classroom materials throughout school ye...
98973     duct duct craft spring inspiration children sp...
52282     discovering our best selves education nurturin...
46375     techies training everyday students interact te...
83528     reading writing technology i teach six amazing...
36468     literacy classroom materials everyday students...
36358     literacy centers i half day pre k i two sets s...
39438     bilingual spanish books elementary library our...
72117     coming soon after school photography club each...
2521      supplies support struggling readers my student...
                                ...
65527     creating an environment for all learners i wor...
24226     beginning teacher eager students endless possi...
35609     a rug reading meeting love sing create move le...
57692     family time during summer break pt 2 our schoo...
96905     learning together 2nd grade my students live o...
27437     art4healing project expressing emotions my 6th...
86437     a clean place sit learn i work wonderful group...
64442     nebraska golden sower nominated books as teach...
60130     flexible seating personalized learning i incre...
61773     exploring literature with graphic toon books i...
83452     em power reading every child deserves champion...
78852     educating young while having fun my students l...
62763     help headphones more our school encountered gr...
98383     academic achievement through chess yes motivat...
108896    the mind body connection although physical edu...
5403      headphones help students hear reach higher suc...
18892     learning in the 21st century my first graders ...
56589     wiggle room my classroom revolving door they e...
21335     support computer science computer graphics cla...
41604     growing independent readers each day students ...
11368     making math fun i teach 17 amazing students ti...
32881     student access missed lessons my students ofte...
```

```
84022     college signing day event our students come mu...
106793    3rd grade flexible seating my students year lo...
27376     learning color i teach first grade title i sch...
87154     nanakuli football projection screen our day st...
14678     operation organization my students range age f...
39096     bringing agriculture sustainability classroom ...
87881     cricket cutting machine needed i teach many di...
78306     news kids my first graders eager learn world a...
Name: total_txt, Length: 109248, dtype: object
```

vectoriziing Categorical data

In [27]:

```
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 will be added in fit and transform will take care of new item. It gives unk
        """
        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 U
        :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 [28]:

```
X_train['teacher_prefix'].fillna(value="Mrs.", inplace=True)
X_cv['teacher_prefix'].fillna(value="Mrs.", inplace=True)
X_test['teacher_prefix'].fillna(value="Mrs.", inplace=True)
vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['teacher_prefix'].values)
teacher_prefix_categories_one_hot_train = vectorizer.transform(X_train['teacher_prefix'].va
teacher_prefix_categories_one_hot_cv = vectorizer.transform(X_cv['teacher_prefix'].values)
teacher_prefix_categories_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].valu

vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['school_state'].values)
school_state_categories_one_hot_train = vectorizer.transform(X_train['school_state'].values
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_state'].values)

vectorizer = LabelEncoderExt()
vectorizer.fit(['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2'])
project_grade_categories_one_hot_train = vectorizer.transform(X_train['project_grade_categc
project_grade_categories_one_hot_cv = vectorizer.transform(X_cv['project_grade_category'].v
project_grade_categories_one_hot_test = vectorizer.transform(X_test['project_grade_category

vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['clean_categories'].values)
categories_one_hot_train = vectorizer.transform(X_train['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)

vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['clean_subcategories'].values)
sub_categories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)
```

1.5 Preparing data for models

In [29]:

```
project_data.columns
```

Out[29]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'project_grade_category', 'clean_categories', 'clean_subcategories',
       'clean_titles', 'clean_essays', 'total_txt'],
      dtype='object')
```

a) Price

In [30]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [31]:

```
# join two dataframes in python:
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
```

In [32]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikitlearn.org/stable/modules/generated/sklearn.preproc
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standarddevi
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
price_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_train
# Now standardize the data with above maen and variance.
price_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_test
# Now standardize the data with above maen and variance.
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_cv
```

Out[32]:

```
array([[0.03504982],
       [0.0277376 ],
       [0.02593731],
       ...,
       [0.02644139],
       [0.02086946],
       [0.00993465]])
```

In [33]:

```
print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041, 2)
(24155, 1) (24155, 2)
(36052, 1) (36052, 2)
```

b) Quantity

In [34]:

```
price_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
quantity_train = price_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_train
# Now standardize the data with above maen and variance.
quantity_cv = price_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_cv
# Now standardize the data with above maen and variance.
quantity_test = price_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
quantity_test
```

Out[34]:

```
array([[0.00111235],
       [0.06562848],
       [0.01334816],
       ...,
       [0.03225806],
       [0.02224694],
       [0.01001112]])
```

In [35]:

```
print("After vectorizations")
print(quantity_train.reshape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
```

```
After vectorizations
<built-in method reshape of numpy.ndarray object at 0x7f113e33de40> (49041,
2)
(24155, 1) (24155, 2)
(36052, 1) (36052, 2)
```

c) Number of Projects previously proposed by Teacher

In [36]:

```
price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
prev_projects_train = price_scalar.transform(X_train['teacher_number_of_previously_posted_p
prev_projects_train
# Now standardize the data with above maen and variance.
prev_projects_cv = price_scalar.transform(X_cv['teacher_number_of_previously_posted_project
prev_projects_cv
# Now standardize the data with above maen and variance.
prev_projects_test = price_scalar.transform(X_test['teacher_number_of_previously_posted_pro
prev_projects_test
```

Out[36]:

```
array([[0.        ],
       [0.        ],
       [0.        ],
       ...,
       [0.        ],
       [0.        ],
       [0.00228833]])
```

In [37]:

```
print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041, 2)
(24155, 1) (24155, 2)
(36052, 1) (36052, 2)
```

In [38]:

```
rem_input_train = np.concatenate((price_train,prev_projects_train,quantity_train),axis=1)
rem_input_cv = np.concatenate((price_cv,prev_projects_cv,quantity_cv),axis=1)
rem_input_test = np.concatenate((price_test,prev_projects_test,quantity_test),axis=1)
```

Text Features : padding

In [39]:

```
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
def padded(encoded_docs):
    max_length = 600
    padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
    return padded_docs
```

In [40]:

```
t = Tokenizer()
t.fit_on_texts(X_train['total_txt'])
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_docs1 = t.texts_to_sequences(X_train['total_txt'])
total_txt_train = padded(encoded_docs1)
encoded_docs2 = t.texts_to_sequences(X_cv['total_txt'])
total_txt_cv = padded(encoded_docs2)
encoded_docs3 = t.texts_to_sequences(X_test['total_txt'])
total_txt_test = padded(encoded_docs3)
```

In [41]:

```
print(total_txt_train.shape)
print(total_txt_cv.shape)
print(total_txt_test.shape)
```

```
(49041, 600)
(24155, 600)
(36052, 600)
```

In [42]:

```
embedding_matrix = np.zeros((vocab_size,300))
for word, i in t.word_index.items():
    embedding_vector = gv.get(word)
    if embedding_vector is not None:
    # words not found in embedding index will be all-zeros.
       embedding_matrix[i] = embedding_vector
```

Tokenization : Categorical features

In [43]:

```
#https://medium.com/@davidheffernan_99410/an-introduction-to-using-categorical-embeddings-e
cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories","cl
cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = X_train[cat].nunique()
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)
```

Model 1

In [52]:

```
import keras
from tensorflow.keras.callbacks import TensorBoard
from keras.regularizers import l2
from keras.layers import SpatialDropout1D, LSTM, BatchNormalization,concatenate,Flatten,Emb
from keras.models import Sequential
from keras import Model,Input
from keras.layers import LeakyReLU
from keras.layers import Reshape,Concatenate
import keras.backend as K
K.clear_session()
ins = []
concat = []
```

Text layers

In [53]:

```
text_input = Input(shape=(600,), name = "text_input")
# max_length = 150 ---->max length of sentence
ins.append(text_input)
e1 = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=600,trainable=Fals

l1= LSTM(128,kernel_initializer=keras.initializers.he_normal(seed=None),recurrent_dropout=0
l1= LeakyReLU(alpha = 0.3)(l1)
f1= Flatten()(l1)
concat.append(f1)
```

Categoraical Layers

In [54]:

```
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)
```

Numerical Layers

In [55]:

```
rem_input = Input(shape=(3,),name="rem_input")
ins.append(rem_input)
rem_dense = Dense(32,activation='relu',kernel_initializer=keras.initializers.he_normal(seed
concat.append(rem_dense)
```

In [56]:

```
#AUC score
import tensorflow as tf
from sklearn.metrics import roc_auc_score

def auc( y_true, y_pred ) :
    score = tf.py_func( lambda y_true, y_pred : roc_auc_score( y_true, y_pred, average='mac
                        [y_true, y_pred],
                        'float32',
                        stateful=True,
                        name='sklearnAUC' )
    return score
```

In [57]:

```
from time import time
x = Concatenate()(concat)

x = Dense(128,activation='relu',kernel_initializer=keras.initializers.he_normal(seed=None),
x = Dropout(0.5)(x)
x = Dense(64,activation='relu',kernel_initializer=keras.initializers.he_normal(seed=None),k
x = Dropout(0.5)(x)
x = BatchNormalization()(x)
x = Dense(32,activation='relu',kernel_initializer=keras.initializers.he_normal(seed=None),k
x = Dropout(0.5)(x)
output = Dense(2, activation = 'softmax')(x)

# create model with seven inputs
model1 = Model(ins , output)
tensorboard = TensorBoard(log_dir='logs/{}'.format(time()))

model1.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=0.0006,c

model1.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected t
o
==================================================================================================
text_input (InputLayer)         (None, 600)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 600, 300)     12913500    text_input
[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 600, 128)     219648      embedding_1
[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
__________________________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 600, 128)     0           lstm_1[0]
[0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 3)         18          teacher_pre
fix[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1, 26)        1352        school_stat
e[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 1, 3)         15          project_gra
de_category[0][0]
__________________________________________________________________________________________________
embedding_5 (Embedding)         (None, 1, 26)        1326        clean_categ
ories[0][0]
__________________________________________________________________________________________________
embedding_6 (Embedding)         (None, 1, 50)        19050       clean_subca
tegories[0][0]
__________________________________________________________________________________________________
rem_input (InputLayer)          (None, 3)            0
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 76800)        0           leaky_re_lu
_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]
__________________________________________________________________________________________________
rem_dense (Dense)               (None, 32)           128         rem_input
[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 76940)        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]
                                                                 rem_dense
```

```
[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          9848448     concatenate
_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_1[0]
[0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 64)           8256        dropout_1
[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           dense_2[0]
[0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 64)           256         dropout_2
[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 32)           2080        batch_norma
lization_1[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 32)           0           dense_3[0]
[0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 2)            66          dropout_3
[0][0]
==================================================================================================
Total params: 23,014,143
Trainable params: 10,100,515
Non-trainable params: 12,913,628
__________________________________________________________________________________________________
```

Visualize the Model

In [58]:

```
#https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/
from keras.utils.vis_utils import plot_model
plot_model(model1, to_file='model1.png', show_shapes=True, show_layer_names=True)
```

In [59]:

```
roject_grade_category': project_grade_categories_one_hot_train,'clean_categories': categori
```

```
ghts_copy.best.hdf5
Epoch 15/30
49041/49041 [==============================] - 437s 9ms/step - loss: 0.429
5 - auc: 0.7147 - val_loss: 0.4366 - val_auc: 0.7139

Epoch 00015: val_auc improved from 0.70883 to 0.71389, saving model to wei
ghts_copy.best.hdf5
Epoch 16/30
49041/49041 [==============================] - 440s 9ms/step - loss: 0.425
0 - auc: 0.7234 - val_loss: 0.4295 - val_auc: 0.7185

Epoch 00016: val_auc improved from 0.71389 to 0.71852, saving model to wei
ghts_copy.best.hdf5
Epoch 17/30
49041/49041 [==============================] - 434s 9ms/step - loss: 0.418
8 - auc: 0.7327 - val_loss: 0.4464 - val_auc: 0.7218

Epoch 00017: val_auc improved from 0.71852 to 0.72183, saving model to wei
ghts_copy.best.hdf5
Epoch 18/30
```

In [60]:

```
from keras.models import load_model
best_model1 = load_model('weights_copy.best.hdf5',custom_objects={"auc":auc})
```

In [61]:

```
result1 = best_model1.evaluate({'text_input': total_txt_test, 'school_state': school_state_
```

```
36052/36052 [==============================] - 95s 3ms/step
```

In [62]:

```
print("{} of test data {}". format(best_model1.metrics_names[0],result1[0]))
print("{} of test data {}". format(best_model1.metrics_names[1],result1[1]))
```

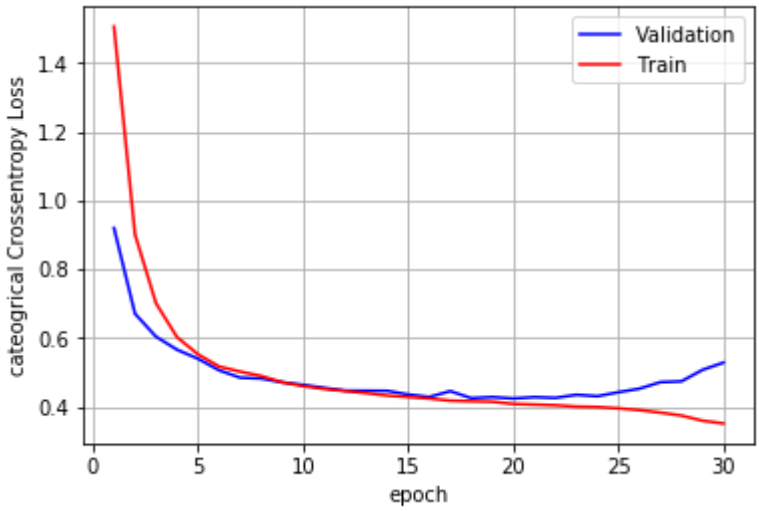
```
loss of test data 0.4223794806159642
auc of test data 0.738992459062703
```

In [63]:

```
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation")
    ax.plot(x, ty, 'r', label="Train")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [65]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('cateogrical Crossentropy Loss')
x = list(range(1,30+1))
vy = history_1.history['val_loss']
ty = history_1.history['loss']
plt_dynamic(x, vy, ty, ax)
```

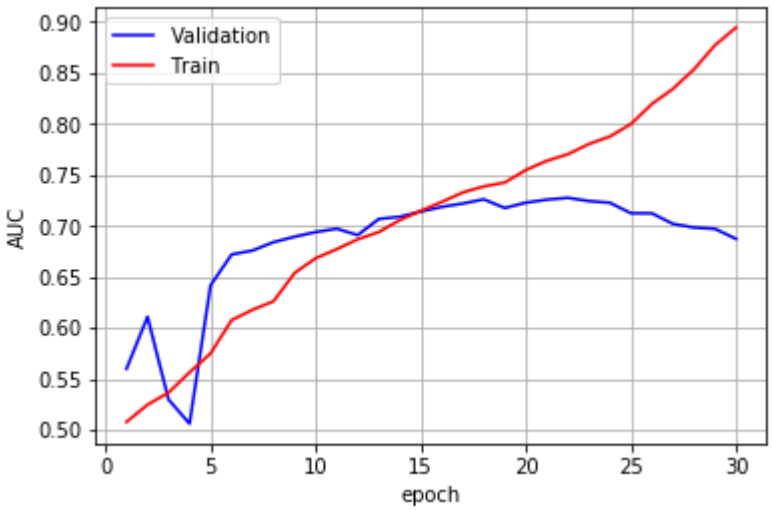


In [66]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('AUC')

# list of epoch numbers
x = list(range(1,30+1))

vy = history_1.history['val_auc']
ty = history_1.history['auc']
plt_dynamic(x, vy, ty, ax)
```



the model got overfitted at epoch 30 i would recommend the epoch to be in range 25

Model 2

In [88]:

```
vectorizer = TfidfVectorizer(min_df=10,max_features=10000) #Defining TFIDF with min_df=10
imp_tf = vectorizer.fit(X_train['total_txt'].values)
idf_val = vectorizer.idf_
```

In [89]:

```
df = pd.DataFrame(idf_val, columns= ["idf"])
df.head()
```

Out[89]:

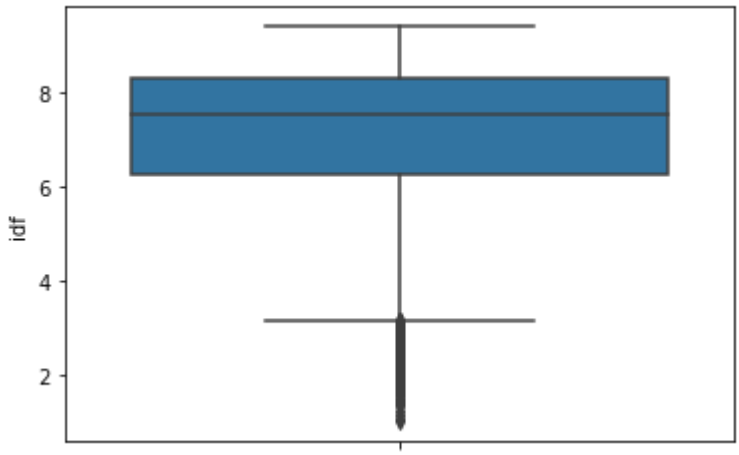
	idf
0	7.081933
1	5.953994
2	8.855993
3	4.477922
4	3.815629

In [90]:

```
sns.boxplot(y = "idf", data = df )
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f10582d7048>
```



In [91]:

```
print("The 25 percentile of idf score is :", np.percentile(vectorizer.idf_,[25]))
print("The 75 percentile of idf score is :",np.percentile(vectorizer.idf_,[75]))
print("The 0 percentile of idf score is :", np.percentile(vectorizer.idf_,[0]))
print("The 100 percentile of idf score is :",np.percentile(vectorizer.idf_,[100]))
```

```
The 25 percentile of idf score is : [6.23975072]
The 75 percentile of idf score is : [8.30392479]
The 0 percentile of idf score is : [1.00722394]
The 100 percentile of idf score is : [9.40253708]
```

In [108]:

```
feature_names = np.asarray(vectorizer.get_feature_names()) # getting all words

index = []
for i in range(len(idf_val)):
    if idf_val[i] >= 3 and idf_val[i] <=10:
        index.append(i)

important_words = []
for i in index:
    important_words.append(feature_names[i])
```

In [128]:

```
# train_data
X_train_imp = []
for sentence in tqdm(X_train['clean_essays']):
    sen = []
    for word in sentence.split():
        if word in important_words:
            sen.append(word)
    X_train_imp.append(' '.join(sen))

#cv_data
X_cv_imp = []
for sentence in tqdm(X_cv['clean_essays']):
    sen = []
    for word in sentence.split():
        if word in important_words:
            sen.append(word)
    X_cv_imp.append(' '.join(sen))

#test_data
X_test_imp = []
for sentence in tqdm(X_test['clean_essays']):
    sen = []
    for word in sentence.split():
        if word in important_words:
            sen.append(word)
    X_test_imp.append(' '.join(sen))
```

```
 91%|▉▉▉▉▉▉▉▉▉ | 44461/49041 [14:04<01:22, 55.37it/s]
 91%|▉▉▉▉▉▉▉▉▉ | 44467/49041 [14:04<01:28, 51.49it/s]
 91%|▉▉▉▉▉▉▉▉▉ | 44473/49041 [14:04<01:26, 53.11it/s]
 91%|▉▉▉▉▉▉▉▉▉ | 44479/49041 [14:04<01:27, 51.85it/s]
 91%|▉▉▉▉▉▉▉▉▉ | 44485/49041 [14:05<01:28, 51.61it/s]
 91%|▉▉▉▉▉▉▉▉▉ | 44492/49041 [14:05<01:23, 54.47it/s]
```

In [129]:

```
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
def padded(encoded_docs):
    max_length = 600
    padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
    return padded_docs
```

In [130]:

```
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
#https://stackoverflow.com/posts/51956230/revisions
t = Tokenizer()
t.fit_on_texts(X_train_imp)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_docs = t.texts_to_sequences(X_train_imp)
total_txt_train = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(X_cv_imp)
total_txt_cv = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(X_test_imp)
total_txt_test = padded(encoded_docs)
```

In [131]:

```
print(total_txt_train.shape)
```

```
(49041, 600)
```

In [132]:

```
embedding_matrix = np.zeros((vocab_size,300))
for word, i in t.word_index.items():
    embedding_vector = gv.get(word)
    if embedding_vector is not None:
    # words not found in embedding index will be all-zeros.
       embedding_matrix[i] = embedding_vector
```

In [133]:

```
import keras
from tensorflow.keras.callbacks import TensorBoard
from keras.regularizers import l2
from keras.layers import SpatialDropout1D, LSTM, BatchNormalization,concatenate,Flatten,Emb
from keras.models import Sequential
from keras import Model,Input
from keras.layers import LeakyReLU
from keras.layers import Reshape,Concatenate
import keras.backend as K
K.clear_session()
ins = []
concat = []
```

In [134]:

```
text_input = Input(shape=(600,), name = "text_input")
# max_length = 150 ---->max length of sentence
ins.append(text_input)
e1 = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=600,trainable=Fals

l1= LSTM(128,kernel_initializer=keras.initializers.he_normal(seed=None),recurrent_dropout=0
l1= LeakyReLU(alpha = 0.3)(l1)
f1= Flatten()(l1)
concat.append(f1)
```

In [135]:

```
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 [136]:

```
rem_input =  Input(shape=(3,), name="rem_input")
ins.append(rem_input)
rem_dense = Dense(64, activation='relu',kernel_initializer=keras.initializers.he_normal(see
concat.append(rem_dense)
```

In [137]:

```
from time import time
x = Concatenate()(concat)
x=BatchNormalization()(x)
#x= Dense(1024, activation='relu')(x)
#x= LeakyReLU(alpha = 0.3)(x)
#x= Dropout(0.8)(x)
#x= Dense(512, activation='relu')(x)
#x= LeakyReLU(alpha = 0.3)(x)
#x= Dropout(0.7)(x)
#x=BatchNormalization()(x)
x= Dense(128,kernel_initializer=keras.initializers.he_normal(seed=None),kernel_regularizer=
x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x= Dense(64,kernel_initializer=keras.initializers.he_normal(seed=None),kernel_regularizer=l
x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.4)(x)
#x= Dense(1024)(x)
#x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x=BatchNormalization()(x)
x= Dense(32,kernel_initializer=keras.initializers.he_normal(seed=None),kernel_regularizer=l
x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x=BatchNormalization()(x)
#x= Dense(16,activation='relu',kernel_initializer=keras.initializers.he_normal(seed=None),k
#x= LeakyReLU(alpha = 0.3)(x)
#x= Dropout(0.25)(x)
output=Dense(2, activation='softmax')(x)

model2 = Model(inputs=ins, outputs=output)

tensorboard = TensorBoard(log_dir='logs/{}'.format(time()))
model2.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=0.0006,c


model2.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected t
o
==================================================================================================
text_input (InputLayer)         (None, 600)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 600, 300)     2956800     text_input
[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 600, 128)     219648      embedding_1
[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
____________________________________________________________________________________________________
__________________________
leaky_re_lu_1 (LeakyReLU)       (None, 600, 128)    0           lstm_1[0]
[0]
____________________________________________________________________________________________________
__________________________
embedding_2 (Embedding)         (None, 1, 3)        18          teacher_pre
fix[0][0]
____________________________________________________________________________________________________
__________________________
embedding_3 (Embedding)         (None, 1, 26)       1352        school_stat
e[0][0]
____________________________________________________________________________________________________
__________________________
embedding_4 (Embedding)         (None, 1, 3)        15          project_gra
de_category[0][0]
____________________________________________________________________________________________________
__________________________
embedding_5 (Embedding)         (None, 1, 26)       1326        clean_categ
ories[0][0]
____________________________________________________________________________________________________
__________________________
embedding_6 (Embedding)         (None, 1, 50)       19150       clean_subca
tegories[0][0]
____________________________________________________________________________________________________
__________________________
rem_input (InputLayer)          (None, 3)           0
____________________________________________________________________________________________________
__________________________
flatten_1 (Flatten)             (None, 76800)       0           leaky_re_lu
_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]
____________________________________________________________________________________________________
__________________________
```

```
rem_dense (Dense)               (None, 64)           256         rem_input
[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 76972)        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]
                                                                 rem_dense
[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 76972)        307888      concatenate
_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          9852544     batch_norma
lization_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 128)          0           dense_1[0]
[0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           leaky_re_lu
_2[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 64)           8256        dropout_1
[0][0]
__________________________________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)       (None, 64)           0           dense_2[0]
[0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           leaky_re_lu
_3[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 64)           0           dropout_2
[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 64)           256         dropout_3
[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 32)           2080        batch_norma
lization_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_4 (LeakyReLU)       (None, 32)           0           dense_3[0]
```

```
[0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 32)           0           leaky_re_lu
_4[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 32)           128         dropout_4
[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 2)            66          batch_norma
lization_3[0][0]
==================================================================================================
Total params: 13,369,783
Trainable params: 10,258,847
Non-trainable params: 3,110,936
__________________________________________________________________________________________________
```

In [138]:

```
#https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/
from keras.utils.vis_utils import plot_model
plot_model(model2, to_file='model2.png', show_shapes=True, show_layer_names=True)
```

In [139]:

```
y_train.shape
```

Out[139]:

```
(49041, 2)
```

In [140]:

```
#AUC score
import tensorflow as tf
from sklearn.metrics import roc_auc_score

def auc( y_true, y_pred ) :
    score = tf.py_func( lambda y_true, y_pred : roc_auc_score( y_true, y_pred, average='mac
                        [y_true, y_pred],
                        'float32',
                        stateful=True,
                        name='sklearnAUC' )
    return score
```

In [141]:

```
from keras.callbacks import ReduceLROnPlateau
reduce_lr= ReduceLROnPlateau(monitor='val_loss', factor=0.2,patience=1, min_lr=0.001,verbos
```

In [142]:

```
from keras.callbacks import EarlyStopping
earlystopping = EarlyStopping(monitor='val_loss', patience=2, verbose=1)
```

In [143]:

```
from keras.callbacks import Callback, ModelCheckpoint
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_only=True, m
```

In [144]:

```
callbacks_list = [checkpoint,tensorboard,earlystopping,reduce_lr]
```

In [145]:

```
#model fitting
from keras.callbacks import Callback, ModelCheckpoint
#https://machinelearningmastery.com/check-point-deep-learning-models-keras/
filepath="weights1_copy.best.hdf5"
history_2 = model2.fit({'text_input': total_txt_train, 'school_state': school_state_categor
```

```
Train on 49041 samples, validate on 24155 samples
Epoch 1/20
49041/49041 [==============================] - 514s 10ms/step - loss: 1.9705
- auc: 0.5234 - val_loss: 1.5491 - val_auc: 0.5948

Epoch 00001: val_auc improved from -inf to 0.59476, saving model to weights1
_copy.best.hdf5
Epoch 2/20
49041/49041 [==============================] - 513s 10ms/step - loss: 1.2564
- auc: 0.5423 - val_loss: 1.0480 - val_auc: 0.6358

Epoch 00002: val_auc improved from 0.59476 to 0.63580, saving model to weigh
ts1_copy.best.hdf5
Epoch 3/20
49041/49041 [==============================] - 504s 10ms/step - loss: 0.9483
- auc: 0.5602 - val_loss: 0.8447 - val_auc: 0.6587

Epoch 00003: val_auc improved from 0.63580 to 0.65872, saving model to weigh
ts1_copy.best.hdf5
Epoch 4/20
49041/49041 [==============================] - 506s 10ms/step - loss: 0.8112
- auc: 0.5752 - val_loss: 0.7382 - val_auc: 0.6676

Epoch 00004: val_auc improved from 0.65872 to 0.66762, saving model to weigh
ts1_copy.best.hdf5
Epoch 5/20
49041/49041 [==============================] - 505s 10ms/step - loss: 0.7343
- auc: 0.6059 - val_loss: 0.6823 - val_auc: 0.6733

Epoch 00005: val_auc improved from 0.66762 to 0.67327, saving model to weigh
ts1_copy.best.hdf5
Epoch 6/20
49041/49041 [==============================] - 507s 10ms/step - loss: 0.6929
- auc: 0.6244 - val_loss: 0.6546 - val_auc: 0.6829

Epoch 00006: val_auc improved from 0.67327 to 0.68288, saving model to weigh
ts1_copy.best.hdf5
Epoch 7/20
49041/49041 [==============================] - 505s 10ms/step - loss: 0.6551
- auc: 0.6402 - val_loss: 0.6260 - val_auc: 0.6898

Epoch 00007: val_auc improved from 0.68288 to 0.68979, saving model to weigh
ts1_copy.best.hdf5
Epoch 8/20
49041/49041 [==============================] - 507s 10ms/step - loss: 0.6400
- auc: 0.6469 - val_loss: 0.6207 - val_auc: 0.6940

Epoch 00008: val_auc improved from 0.68979 to 0.69400, saving model to weigh
ts1_copy.best.hdf5
Epoch 9/20
49041/49041 [==============================] - 506s 10ms/step - loss: 0.6126
- auc: 0.6707 - val_loss: 0.5865 - val_auc: 0.7010
```

```
Epoch 00009: val_auc improved from 0.69400 to 0.70104, saving model to weigh
ts1_copy.best.hdf5
Epoch 10/20
49041/49041 [==============================] - 502s 10ms/step - loss: 0.5860
- auc: 0.6795 - val_loss: 0.5684 - val_auc: 0.7000

Epoch 00010: val_auc did not improve from 0.70104
Epoch 11/20
49041/49041 [==============================] - 512s 10ms/step - loss: 0.5666
- auc: 0.6888 - val_loss: 0.5543 - val_auc: 0.7067

Epoch 00011: val_auc improved from 0.70104 to 0.70673, saving model to weigh
ts1_copy.best.hdf5
Epoch 12/20
49041/49041 [==============================] - 513s 10ms/step - loss: 0.5534
- auc: 0.6948 - val_loss: 0.5402 - val_auc: 0.7074

Epoch 00012: val_auc improved from 0.70673 to 0.70737, saving model to weigh
ts1_copy.best.hdf5
Epoch 13/20
49041/49041 [==============================] - 514s 10ms/step - loss: 0.5379
- auc: 0.7037 - val_loss: 0.5350 - val_auc: 0.7022

Epoch 00013: val_auc did not improve from 0.70737
Epoch 14/20
49041/49041 [==============================] - 515s 11ms/step - loss: 0.5261
- auc: 0.7140 - val_loss: 0.5232 - val_auc: 0.7054

Epoch 00014: val_auc did not improve from 0.70737
Epoch 15/20
49041/49041 [==============================] - 514s 10ms/step - loss: 0.5151
- auc: 0.7227 - val_loss: 0.5169 - val_auc: 0.7048

Epoch 00015: val_auc did not improve from 0.70737
Epoch 16/20
49041/49041 [==============================] - 516s 11ms/step - loss: 0.5067
- auc: 0.7302 - val_loss: 0.5150 - val_auc: 0.7048

Epoch 00016: val_auc did not improve from 0.70737
Epoch 17/20
49041/49041 [==============================] - 513s 10ms/step - loss: 0.4991
- auc: 0.7412 - val_loss: 0.5117 - val_auc: 0.6996

Epoch 00017: val_auc did not improve from 0.70737
Epoch 18/20
49041/49041 [==============================] - 510s 10ms/step - loss: 0.4895
- auc: 0.7548 - val_loss: 0.5154 - val_auc: 0.6966

Epoch 00018: val_auc did not improve from 0.70737
Epoch 19/20
49041/49041 [==============================] - 511s 10ms/step - loss: 0.4792
- auc: 0.7652 - val_loss: 0.5122 - val_auc: 0.6944

Epoch 00019: val_auc did not improve from 0.70737
Epoch 00019: early stopping
```

In [151]:

```
from keras.models import load_model
best_model2 = load_model('weights1_copy.best.hdf5',custom_objects={"auc":auc})
```

In [152]:

```
result2 = best_model2.evaluate({'text_input': total_txt_test, 'school_state': school_state_
```

```
36052/36052 [==============================] - 109s 3ms/step
```

In [153]:

```
print("{} of test data {}". format(best_model2.metrics_names[0],result2[0]))
print("{} of test data {}". format(best_model2.metrics_names[1],result2[1]))
```

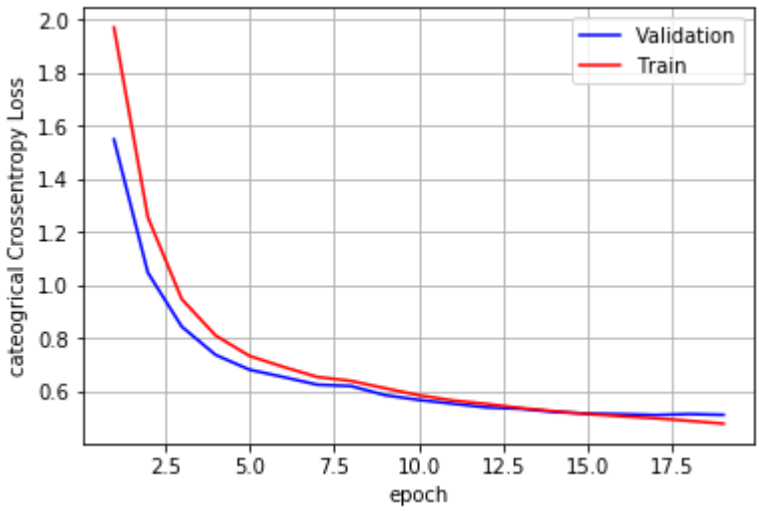
```
loss of test data 0.5361523789265941
auc of test data 0.7194239080686303
```

In [154]:

```
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation")
    ax.plot(x, ty, 'r', label="Train")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [155]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('cateogrical Crossentropy Loss')
x = list(range(1,19+1))
vy = history_2.history['val_loss']
ty = history_2.history['loss']
plt_dynamic(x, vy, ty, ax)
```

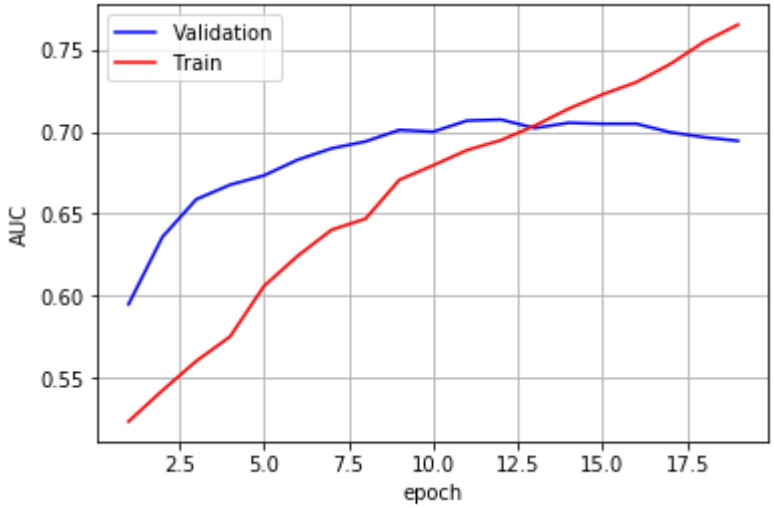


In [156]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('AUC')

# list of epoch numbers
x = list(range(1,19+1))

vy = history_2.history['val_auc']
ty = history_2.history['auc']
plt_dynamic(x, vy, ty, ax)
```



model 3

In [44]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
X_train_teacher_prefiX_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_teacher_prefiX_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_prefiX_ohe = vectorizer.transform(X_test['teacher_prefix'].values)
teacher_prefiX_f=vectorizer.get_feature_names()
print("After vectorizations of teacher_prefix")
print(X_train_teacher_prefiX_ohe.shape, y_train.shape)
print(X_cv_teacher_prefiX_ohe.shape, y_cv.shape)
print(X_test_teacher_prefiX_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())

print("=========================================================")

vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values)
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)
state_f=vectorizer.get_feature_names()
print("After vectorizations of school_state")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("=========================================================")
vectorizer = CountVectorizer()
vectorizer.fit(['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2'])
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)
teacher_grade_f=vectorizer.get_feature_names()
print("After vectorizations of project_grade_category")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())

print("=========================================================")

vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values)
X_train_cat_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_cat_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_cat_ohe = vectorizer.transform(X_test['clean_categories'].values)
teacher_cat_f=vectorizer.get_feature_names()
print("After vectorizations of clean_categories")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_cv_cat_ohe.shape, y_cv.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())

print("=========================================================")

vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values)
X_train_scat_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_scat_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_scat_ohe = vectorizer.transform(X_test['clean_subcategories'].values)
```

```
teacher_scat_f=vectorizer.get_feature_names()
print("After vectorizations of clean_subcategories ")
print(X_train_scat_ohe.shape, y_train.shape)
print(X_cv_scat_ohe.shape, y_cv.shape)
print(X_test_scat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations of teacher_prefix
(49041, 5) (49041, 2)
(24155, 5) (24155, 2)
(36052, 5) (36052, 2)
['dr', 'mr', 'mrs', 'ms', 'teacher']
=========================================================
After vectorizations of school_state
(49041, 51) (49041, 2)
(24155, 51) (24155, 2)
(36052, 51) (36052, 2)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'i
a', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo',
'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'o
r', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv',
'wy']
=========================================================
After vectorizations of project_grade_category
(49041, 4) (49041, 2)
(24155, 4) (24155, 2)
(36052, 4) (36052, 2)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
=========================================================
After vectorizations of clean_categories
(49041, 9) (49041, 2)
(24155, 9) (24155, 2)
(36052, 9) (36052, 2)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'liter
acy_language', 'math_science', 'music_arts', 'specialneeds', 'warmth']
=========================================================
After vectorizations of clean_subcategories
(49041, 30) (49041, 2)
(24155, 30) (24155, 2)
(36052, 30) (36052, 2)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_governmen
t', 'college_careerprep', 'communityservice', 'earlydevelopment', 'economic
s', 'environmentalscience', 'esl', 'extracurricular', 'financialliteracy',
'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness',
'history_geography', 'literacy', 'literature_writing', 'mathematics', 'musi
c', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 's
ocialsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
```

In [45]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikitlearn.org/stable/modules/generated/sklearn.preproc
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standarddevi
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
X_train_price_norm = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
X_train_price_norm
# Now standardize the data with above maen and variance.
X_test_price_norm = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
X_test_price_norm
# Now standardize the data with above maen and variance.
X_cv_price_norm = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
X_cv_price_norm
```

Out[45]:

```
array([[0.03504982],
       [0.0277376 ],
       [0.02593731],
       ...,
       [0.02644139],
       [0.02086946],
       [0.00993465]])
```

In [46]:

```
price_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
X_train_qty_norm= price_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_train
# Now standardize the data with above maen and variance.
X_cv_qty_norm= price_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_cv
# Now standardize the data with above maen and variance.
X_test_qty_norm= price_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
quantity_test
```

Out[46]:

```
array([[0.00111235],
       [0.06562848],
       [0.01334816],
       ...,
       [0.03225806],
       [0.02224694],
       [0.01001112]])
```

In [47]:

```
price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
X_train_tpp_norm= price_scalar.transform(X_train['teacher_number_of_previously_posted_proje
X_train_tpp_norm
# Now standardize the data with above maen and variance.
X_cv_tpp_norm= price_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].
X_cv_tpp_norm
# Now standardize the data with above maen and variance.
X_test_tpp_norm= price_scalar.transform(X_test['teacher_number_of_previously_posted_project
X_test_tpp_norm
```

Out[47]:

```
array([[0.        ],
       [0.        ],
       [0.        ],
       ...,
       [0.        ],
       [0.        ],
       [0.00228833]])
```

In [48]:

```
from scipy.sparse import hstack
X_tr_rem = hstack((X_train_state_ohe, X_train_teacher_prefiX_ohe, X_train_grade_ohe,X_trair
X_cv_rem = hstack(( X_cv_state_ohe, X_cv_teacher_prefiX_ohe, X_cv_grade_ohe,X_cv_scat_ohe,X
X_te_rem = hstack((X_test_state_ohe, X_test_teacher_prefiX_ohe, X_test_grade_ohe,X_test_sca
print("Final Data matrix")
print(X_tr_rem.shape, y_train.shape)
print(X_cv_rem.shape, y_cv.shape)
print(X_te_rem.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 102) (49041, 2)
(24155, 102) (24155, 2)
(36052, 102) (36052, 2)
==============================================================================
========================
```

In [49]:

```
X_tr_rem_reshape = np.array(X_tr_rem).reshape(49041,102,1)
X_cv_rem_reshape = np.array(X_cv_rem).reshape(24155, 102,1)
X_test_rem_reshape = np.array(X_te_rem).reshape(36052, 102,1)
```

In [50]:

```
y_train.shape
```

Out[50]:

```
(49041, 2)
```

In [51]:

```
max_length = 400
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
def padded(encoded_docs):
    max_length = 400
    padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
    return padded_docs
```

In [53]:

```
#https://stackoverflow.com/posts/51956230/revisions
t = Tokenizer()
t.fit_on_texts(X_train['total_txt'])
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_docs = t.texts_to_sequences(X_train['total_txt'])
total_txt_train = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(X_cv['total_txt'])
total_txt_cv = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(X_test['total_txt'])
total_txt_test = padded(encoded_docs)
```

In [54]:

```
embedding_matrix = np.zeros((vocab_size,300))
for word, i in t.word_index.items():
    embedding_vector = gv.get(word)
    if embedding_vector is not None:
    # words not found in embedding index will be all-zeros.
       embedding_matrix[i] = embedding_vector
```

In [71]:

```
import keras
from tensorflow.keras.callbacks import TensorBoard
from keras.regularizers import l2
from keras.layers import SpatialDropout1D, LSTM, BatchNormalization,concatenate,Flatten,Emb
from keras.models import Sequential
from keras import Model,Input
from keras.layers import LeakyReLU
from keras.layers import Reshape,Concatenate
import keras.backend as K
K.clear_session()
ins = []
concat = []
```

In [72]:

```
text_input = Input(shape=(400,), name = "text_input")
# max_length = 150 ---->max length of sentence

e1 = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=400)(text_input)

l1= LSTM(128,activation = "relu",dropout=0.5,kernel_regularizer=l2(0.0001),kernel_initializ
f1= Flatten()(l1)

rem = Input(shape=(X_tr_rem.shape[1],1), name="rem")
rem_conv1 = Conv1D(128, 3,kernel_initializer='he_normal')(rem)
act1= LeakyReLU(alpha = 0.3)(rem_conv1)
max_pool =MaxPooling1D(3)(act1)
f2= Flatten()(max_pool)

x = concatenate([f1,f2])
x= Dense(32,kernel_regularizer=l2(0.0001),kernel_initializer='he_normal')(x)
x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x= Dense(16, activation='relu')(x)
output=Dense(2, activation='softmax')(x)
model3 = Model(inputs=[text_input,rem], outputs=output)
model3.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected t
o
==================================================================================================
rem (InputLayer)                (None, 102, 1)       0
__________________________________________________________________________________________________
text_input (InputLayer)         (None, 400)          0
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 100, 128)     512         rem[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 400, 300)     12899700    text_input
[0][0]
__________________________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 100, 128)     0           conv1d_1[0]
[0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 400, 128)     219648      embedding_1
[0][0]
__________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 33, 128)      0           leaky_re_lu
_1[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 51200)        0           lstm_1[0]
[0]
__________________________________________________________________________________________________
```

```
flatten_2 (Flatten)             (None, 4224)         0           max_pooling
1d_1[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 55424)        0           flatten_1
[0][0]
                                                                 flatten_2
[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 32)           1773600     concatenate
_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 32)           0           dense_1[0]
[0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 32)           0           leaky_re_lu
_2[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 16)           528         dropout_1
[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 2)            34          dense_2[0]
[0]
==================================================================================================
Total params: 14,894,022
Trainable params: 14,894,022
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [73]:

```
from keras.callbacks import ReduceLROnPlateau
reduce_lr= ReduceLROnPlateau(monitor='val_loss', factor=0.2,patience=1, min_lr=0.001,verbos
```

In [74]:

```
from keras.callbacks import EarlyStopping
earlystopping = EarlyStopping(monitor='val_loss', patience=2, verbose=1)
```

In [75]:

```
from keras.callbacks import Callback, ModelCheckpoint
filepath="weights3_copy.best.hdf5"
checkpoint3 = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_only=True,
```

In [76]:

```
from time import time
tensorboard = TensorBoard(log_dir='logs/{}'.format(time()))
callbacks_list = [checkpoint3,tensorboard,earlystopping,reduce_lr]
```

In [77]:

```
#https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/
from keras.utils.vis_utils import plot_model
plot_model(model3, to_file='model3.png', show_shapes=True, show_layer_names=True)
```

In [78]:

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

def auc( y_true, y_pred ) :
    score = tf.py_func( lambda y_true, y_pred : roc_auc_score( y_true, y_pred, average='mac
                        [y_true, y_pred],
                        'float32',
                        stateful=True,
                        name='sklearnAUC' )
    return score
```

In [79]:

```
model3.compile(loss='categorical_crossentropy', optimizer='adam', metrics=[auc])
```

In [80]:

```
history_3= model3.fit({'text_input': total_txt_train, 'rem':X_tr_rem_reshape},y_train,
          epochs=25, batch_size=512,verbose=1, validation_data=({'text_input': total_txt_cv
```

```
Train on 49041 samples, validate on 24155 samples
Epoch 1/25
49041/49041 [==============================] - 352s 7ms/step - loss: 0.6236
- auc: 0.5554 - val_loss: 0.5424 - val_auc: 0.5763

Epoch 00001: val_auc improved from -inf to 0.57625, saving model to weights3
_copy.best.hdf5
Epoch 2/25
49041/49041 [==============================] - 349s 7ms/step - loss: 0.4710
- auc: 0.6312 - val_loss: 0.5527 - val_auc: 0.7009

Epoch 00002: val_auc improved from 0.57625 to 0.70088, saving model to weigh
ts3_copy.best.hdf5

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 3/25
49041/49041 [==============================] - 353s 7ms/step - loss: 0.4310
- auc: 0.7078 - val_loss: 0.4670 - val_auc: 0.7195

Epoch 00003: val_auc improved from 0.70088 to 0.71953, saving model to weigh
ts3_copy.best.hdf5
Epoch 4/25
49041/49041 [==============================] - 348s 7ms/step - loss: 0.4074
- auc: 0.7391 - val_loss: 0.4887 - val_auc: 0.7288

Epoch 00004: val_auc improved from 0.71953 to 0.72880, saving model to weigh
ts3_copy.best.hdf5

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 5/25
49041/49041 [==============================] - 350s 7ms/step - loss: 0.3908
- auc: 0.7591 - val_loss: 0.4498 - val_auc: 0.7307

Epoch 00005: val_auc improved from 0.72880 to 0.73065, saving model to weigh
ts3_copy.best.hdf5
Epoch 6/25
49041/49041 [==============================] - 343s 7ms/step - loss: 0.3754
- auc: 0.7808 - val_loss: 0.4462 - val_auc: 0.7321

Epoch 00006: val_auc improved from 0.73065 to 0.73213, saving model to weigh
ts3_copy.best.hdf5
Epoch 7/25
49041/49041 [==============================] - 344s 7ms/step - loss: 0.3590
- auc: 0.8026 - val_loss: 0.4369 - val_auc: 0.7266

Epoch 00007: val_auc did not improve from 0.73213
Epoch 8/25
49041/49041 [==============================] - 341s 7ms/step - loss: 0.3453
- auc: 0.8226 - val_loss: 0.4337 - val_auc: 0.7227

Epoch 00008: val_auc did not improve from 0.73213
Epoch 9/25
49041/49041 [==============================] - 349s 7ms/step - loss: 0.3310
- auc: 0.8432 - val_loss: 0.4185 - val_auc: 0.7163
```

```
Epoch 00009: val_auc did not improve from 0.73213
Epoch 10/25
49041/49041 [==============================] - 339s 7ms/step - loss: 0.3131
- auc: 0.8650 - val_loss: 0.4362 - val_auc: 0.7057

Epoch 00010: val_auc did not improve from 0.73213

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 11/25
49041/49041 [==============================] - 344s 7ms/step - loss: 0.2915
- auc: 0.8867 - val_loss: 0.4454 - val_auc: 0.6991

Epoch 00011: val_auc did not improve from 0.73213

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 00011: early stopping
```

In [81]:

```
from keras.models import load_model
best_model3 = load_model('weights3_copy.best.hdf5',custom_objects={"auc":auc})
```

In [82]:

```
result3 = best_model3.evaluate({'text_input': total_txt_test, 'rem':X_test_rem_reshape},y_t
```

```
36052/36052 [==============================] - 59s 2ms/step
```

In [83]:

```
print("{} of test data {}". format(best_model3.metrics_names[0],result3[0]))
print("{} of test data {}". format(best_model3.metrics_names[1],result3[1]))
```

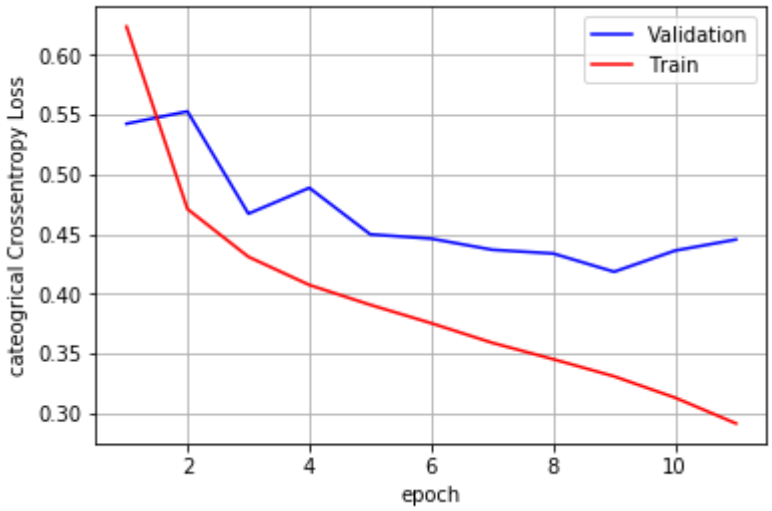
```
loss of test data 0.4403218572152702
auc of test data 0.7462072292747634
```

In [85]:

```
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation")
    ax.plot(x, ty, 'r', label="Train")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [86]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('cateogrical Crossentropy Loss')
x = list(range(1,11+1))
vy = history_3.history['val_loss']
ty = history_3.history['loss']
plt_dynamic(x, vy, ty, ax)
```

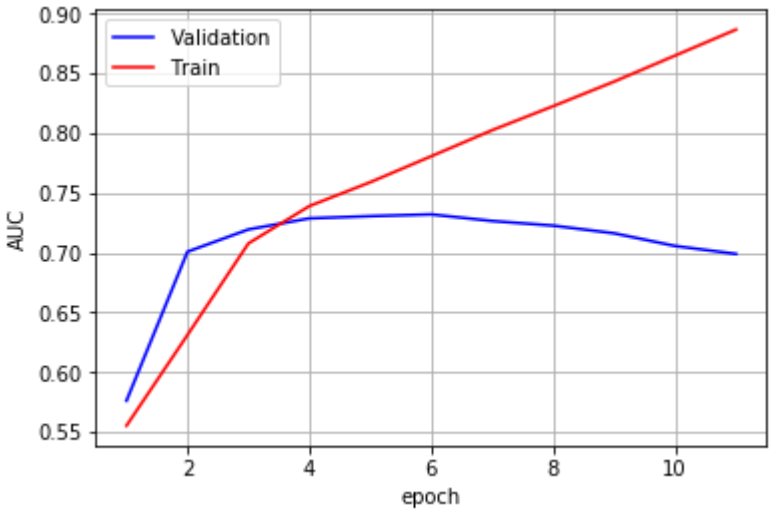


In [87]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('AUC')

# list of epoch numbers
x = list(range(1,11+1))

vy = history_3.history['val_auc']
ty = history_3.history['auc']
plt_dynamic(x, vy, ty, ax)
```



In [157]:

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["model","train_auc","cv_auc","test_auc"]
x.add_row(["model1","0.770","0.727","0.738"])
x.add_row(["model2","0.694","0.707","0.719"])
x.add_row(["model3","0.780 ","0.712","0.746"])

print(x)
```

```
+--------+-----------+--------+----------+
| model  | train_auc | cv_auc | test_auc |
+--------+-----------+--------+----------+
| model1 |   0.770   | 0.727  |  0.738   |
| model2 |   0.694   | 0.707  |  0.719   |
| model3 |   0.780   | 0.712  |  0.746   |
+--------+-----------+--------+----------+
```

In [ ]: