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.utils import class_weight
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
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

import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.initializers import RandomNormal
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Embedding, Dense, Flatten, concate
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, Tensor
```

Splitting data into Train and Test

In [2]:

```
prepeocessed_data = pd.read_csv('preprocessed_data.csv')
prepeocessed_data.head(2)
```

Out[2]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	teacher
0	0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	
1	1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	

2 rows × 21 columns

In [3]:

```
y = prepeocessed_data['project_is_approved'].values
X = prepeocessed_data.drop(['project_is_approved'], axis=1)
X.shape
```

Out[3]:

```
(109248, 20)
```

In [4]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stra
X_test.shape
```

Out[4]:

```
(21850, 20)
```

In [5]:

```
X_train.head(2)
```

Out[5]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	tea
64518	64518	54069	p104194	88fb05940c51dbf6a367a132ad6a9359	
51241	51241	66526	p166289	10d3554fd00e353c1655a88c9bd586c4	

In [6]:

```
y_train
```

Out[6]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [7]:

```
X_test.head(2)
```

Out[7]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	tea
4011	4011	57331	p124796	cde78a5e5129129cf99f58fd7a2da1d7	
89758	89758	151793	p037937	a1a0a7dcd9d20051988f507cd353508f	

In [8]:

```
y_test
```

Out[8]:

```
array([1, 1, 1, ..., 1, 0, 0], dtype=int64)
```

In [9]:

```
vectorizer = TfidfVectorizer(min_df=6,use_idf=True)
vectorizer.fit(X_train['essay'])


X_train_essay=vectorizer.transform(X_train['essay'].values)
X_test_essay=vectorizer.transform(X_test['essay'].values)

print(X_train_essay.shape)
print(X_test_essay.shape)
```

```
(87398, 19734)
(21850, 19734)
```

In [10]:

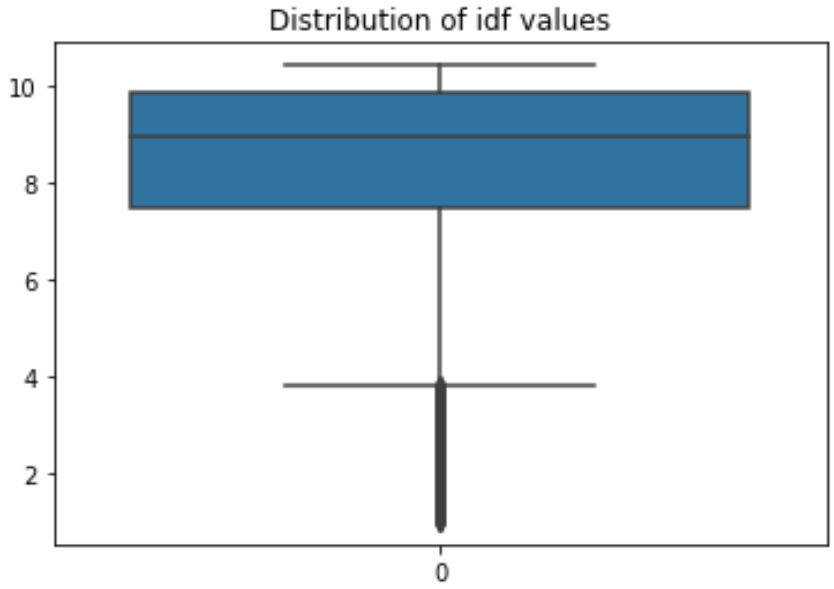
```
idf = vectorizer.idf_
idf_values= dict(zip(vectorizer.get_feature_names(), idf))
df=pd.DataFrame(idf_values.items())
df=df.sort_values(by=1)
df.head()
```

Out[10]:

	0	1
17963	to	1.000137
1019	and	1.000504
17750	the	1.002509
17090	students	1.007765
12385	of	1.011033

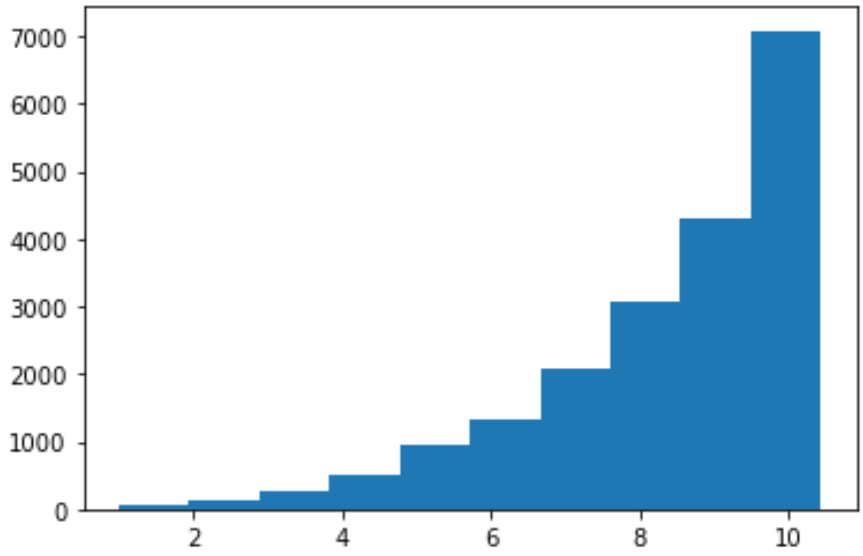
In [11]:

```
sns.boxplot(data=df[1]).set_title('Distribution of idf values')
plt.show()
```



In [12]:

```
plt.hist(df[1])
plt.show()
```



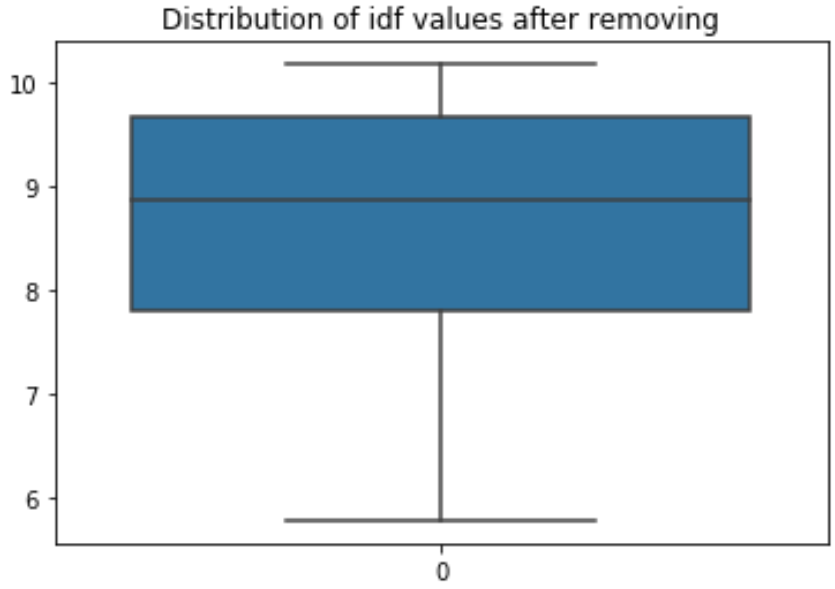
In [13]:

```
iqd1=df[1].quantile(0.10)
iqd2=df[1].quantile(0.90)
print(iqd1)
print(iqd2)
```

```
5.771049305938853
10.298797578250738
```

In [14]:

```
df_1 = df[(df[1] > iqd1) & (df[1] < iqd2)]
sns.boxplot(data=df_1[1]).set_title('Distribution of idf values after removir
plt.show()
df_1.shape
```



Out[14]:

```
(15504, 2)
```

In [15]:

```
df_2 = df[(df[1] <= iqd1) | (df[1] >= iqd2)]
remove=list(df_2[0])
len(remove)
```

Out[15]:

```
4230
```

In [16]:

```
def remove_words(data):
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sent in tqdm(data.values):
        sent = ' '.join(e for e in sent.split() if e.lower() not in remove)
        preprocessed_essays.append(sent.lower().strip())
    return preprocessed_essays
```

In [17]:

```
X_train['essay']=remove_words(X_train['essay'])
X_test['essay']=remove_words(X_test['essay'])
```

```
100%|████████████████████████████████████████████████████████████| 87398/87398 [04:43<00:00, 308.24it/s]
100%|████████████████████████████████████████████████████████████| 21850/21850 [01:10<00:00, 309.14it/s]
```

In [18]:

```
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
```

```
929it [00:00, 9222.68it/s]

Loading Glove Model

1917495it [03:26, 9296.35it/s]

Done. 1917495  words loaded!
```

In [19]:

```
t = Tokenizer()
t.fit_on_texts(X_train['essay'].values)
vocab_size = len(t.word_index) + 1
# integer encode the documents
X_train_encoded_docs = t.texts_to_sequences(X_train['essay'].values)
X_test_encoded_docs = t.texts_to_sequences(X_test['essay'].values)

# pad documents to a max length of 300 words
max_length = 300
X_train_padded_docs = pad_sequences(X_train_encoded_docs, maxlen=max_length,
X_test_padded_docs = pad_sequences(X_test_encoded_docs, maxlen=max_length, pa

embedding_matrix = np.zeros((vocab_size, max_length))

for word, i in tqdm(t.word_index.items()):
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

print(X_train_padded_docs.shape)
print(X_test_padded_docs.shape)
```

```
100%|████████████████████████████████████████████████████████████
████████████████| 61238/61238 [00:00<00:00, 417636.77it/s]

(87398, 300)
(21850, 300)
```

In [20]:

```
print(vocab_size)
print(embedding_matrix.shape)
print(embedding_matrix[1])
```

```
61239
(61239, 300)
[-3.5835e-02  7.7844e-01 -5.1806e-01  8.0682e-02 -1.3173e-0
1 -2.8606e-01
 -4.2485e+00  8.1827e-01  2.4034e-01 -6.9057e-01  2.1556e-0
1 -1.2434e-01
 -5.4229e-01 -1.5138e-01 -1.3591e-01 -5.3480e-01  3.4595e-0
1 -2.2926e-01
 -1.3789e-02  4.7816e-02 -4.1427e-02 -4.1058e-01  1.0172e-0
1 -9.7093e-02
 -6.4646e-04 -2.6877e-02 -2.3473e-01 -6.0190e-02 -3.1410e-0
1 -5.6240e-01
 -7.9395e-02 -2.2116e-01  9.8952e-02  8.9909e-02 -6.8699e-0
2  3.0787e-01
  3.5724e-01  2.7748e-01  2.1167e-01 -2.0939e-02 -4.8188e-0
1 -1.4981e-01
 -1.9693e-01 -1.9514e-01 -2.9672e-02 -2.8620e-01  5.9519e-0
2 -1.5534e-01
 -4.1982e-02  3.1305e-01 -1.0834e-01  7.4492e-01 -2.6733e-0
1  9.0418e-02
  3.2783e-01 -3.5072e-01  1.9352e-03 -1.6018e-01  3.5431e-0
1 -6.6866e-02
 -9.4903e-02 -2.6866e-02 -4.4347e-01  1.3844e-01  1.4952e-0
1  3.4483e-01
  1.2546e-01  4.6310e-01 -5.8689e-03 -4.1446e-01  1.2612e-0
1  1.3602e-01
 -4.3715e-01  9.6268e-02 -1.8979e-01 -7.5418e-01  3.5777e-0
1 -1.7479e-02
 -2.1907e-01 -1.8382e-01 -2.7002e-01 -5.4582e-01 -4.5421e-0
1 -5.4994e-01
 -4.3079e-01 -1.1863e-01 -4.7369e-01 -9.9825e-02 -1.4261e-0
1 -8.1525e-02
  3.3486e-01 -4.1460e-01 -7.9636e-02  3.0339e-01 -4.2575e-0
1 -3.4956e-01
 -2.7357e+00 -7.4104e-01  2.1630e-01  4.2723e-01  3.5431e-0
2 -5.4640e-02
  7.0669e-01 -2.9739e-01  8.3191e-03 -1.9281e-01  3.2391e-0
1  8.7548e-03
 -1.6015e-01 -5.9082e-02  1.7503e-01  3.5089e-01 -1.3540e-0
2 -3.3457e-01
 -5.1774e-02  1.5690e-02  1.0290e+00  4.4273e-02  4.4906e-0
1 -2.9036e-01
 -7.8684e-01 -6.0001e-02  1.1784e-01 -1.1940e-01  2.0565e-0
1 -3.0654e-01
 -1.0883e-01 -7.6856e-02 -2.7208e-02 -3.2139e-03 -1.9201e-0
1  1.3827e-01
```

```
  1.5239e-01 -8.5574e-02  2.5771e-01  3.5583e-02  2.5709e-0
1 -1.3264e-01
  6.4585e-01  5.3900e-01  9.3380e-02  1.7612e-01  4.5444e-0
1 -3.0252e-01
 -5.1751e-01 -7.7072e-03  5.2078e-01 -2.2996e-01  5.1313e-0
1  6.1850e-01
 -2.7100e-01  1.8719e-01  3.4660e-01 -2.7276e-01  3.4786e-0
1 -3.1713e-01
  1.3038e-01  2.1235e-02 -2.5821e-01 -1.4553e-01  6.5942e-0
1 -1.9469e-01
 -1.7672e-01 -1.7879e-01 -2.9119e-01  2.7390e-01 -7.4949e-0
2 -3.5214e-01
  1.2527e-01 -1.5150e-01  1.6561e-01 -3.5536e-01  2.6169e-0
1 -6.6952e-01
 -3.7627e-02  2.4556e-01 -4.7338e-01 -1.9609e-01 -1.6514e-0
1  1.8519e-01
  7.4366e-01  9.9546e-02 -3.0843e-01  8.2241e-02 -1.4698e-0
1 -5.3421e-01
 -6.1243e-03  5.7767e-01  3.4576e-01 -1.6245e-01  6.2227e-0
2 -2.7400e-01
 -1.0936e-01  1.7974e-01  3.2881e-01 -9.0290e-04  7.2287e-0
2  1.1748e-01
  2.5142e-01  6.2339e-02  4.4903e-01 -9.4289e-04  1.8164e-0
1 -2.9528e-01
  1.7274e-01  4.1538e-01 -1.9120e-01  1.4755e-01  3.9594e-0
1 -1.6013e-01
  6.4232e-02 -3.2972e-01  2.0030e-01  3.1493e-01 -4.3744e-0
1 -1.3939e-01
 -1.0015e-01 -2.0848e-01 -2.2608e-01 -4.8169e-02  2.0638e-0
1  5.1249e-01
 -3.4100e-01 -4.2260e-03 -3.4638e+00 -1.2878e-01 -4.7151e-0
1 -4.2011e-01
 -4.0580e-02  4.2346e-02  1.1615e-01 -4.1935e-02  3.8201e-0
2 -2.4746e-01
 -9.3521e-02  4.4521e-01 -4.4714e-02  3.2429e-01 -1.3014e-0
1 -3.9150e-01
 -4.6100e-01 -7.0122e-02 -3.5223e-01  3.8686e-01 -2.5869e-0
1  4.7497e-01
  7.6472e-02  8.5433e-03 -4.0470e-01  6.6155e-01 -2.3044e-0
1 -2.1724e-02
 -2.1885e-02 -5.4206e-04  6.5346e-03 -7.4982e-02 -5.9198e-0
1 -6.1204e-01
 -1.4000e-01  1.9463e-01 -1.7722e-01 -5.1057e-01  4.0299e-0
1  2.6750e-01
 -2.9331e-01 -1.8907e-01 -3.7621e-01 -4.1790e-01  5.4875e-0
1  7.0255e-02
  8.0564e-01 -3.8410e-01 -4.2401e-01 -4.1384e-01  4.3875e-0
1 -2.9252e-01
 -9.1183e-02  2.2039e-01 -1.8372e-01 -4.1012e-01  6.2847e-0
1  2.1983e-01
 -7.9124e-02  1.9266e-02  8.5543e-01 -1.3378e-01  6.0141e-0
1  6.7718e-01
 -3.3309e-01 -2.5610e-01  8.4727e-02  1.0459e-01 -2.5359e-0
```

```
1 -1.2002e-01
 -3.8965e-01 -2.8780e-01  3.6703e-03  2.2321e-02 -3.1591e-0
1 -3.5608e-01]
```

1.4 Encoding Categorical and Numerical features

In [21]:

```
from sklearn.preprocessing import LabelEncoder
import numpy as np


class LabelEncoderExt(object):
    def __init__(self):
        """
        It differs from LabelEncoder by handling new classes and providing a
        Unknown will be added in fit and transform will take care of new item
        """
        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 unk
        :param data_list: A list of string
        :return: self
        """
        self.label_encoder = self.label_encoder.fit(list(data_list) + ['Unkno
        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
        :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

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

1.4.1 encoding categorical features: clean_categories

In [22]:

```
from sklearn.preprocessing import LabelEncoder
vectorizer_cat = LabelEncoderExt()
vectorizer_cat.fit(X_train['clean_categories'].values) # fit has to happen or

X_train_cc_ohe = vectorizer_cat.transform(X_train['clean_categories'].values)
X_test_cc_ohe = vectorizer_cat.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cc_ohe.shape, y_train.shape)
print(X_test_cc_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.2 encoding categorical features: clean_subcategories

In [23]:

```
vectorizer_subcat = LabelEncoderExt()
vectorizer_subcat.fit(X_train['clean_subcategories'].values) # fit has to hap

X_train_csc_ohe = vectorizer_subcat.transform(X_train['clean_subcategories'].
X_test_csc_ohe = vectorizer_subcat.transform(X_test['clean_subcategories'].va

print("After vectorizations")
print(X_train_csc_ohe.shape, y_train.shape)
print(X_test_csc_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.3 encoding categorical features: school_state

In [24]:

```
vectorizer_school_state = LabelEncoderExt()
vectorizer_school_state.fit(X_train['school_state'].values)

X_train_state_ohe = vectorizer_school_state.transform(X_train['school_state']
X_test_state_ohe = vectorizer_school_state.transform(X_test['school_state'].v

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.4 encoding categorical features: teacher_prefix

In [25]:

```
vectorizer_prefix = LabelEncoderExt()
vectorizer_prefix.fit(X_train['teacher_prefix'].values)

X_train_teacher_ohe = vectorizer_prefix.transform(X_train['teacher_prefix'].v
X_test_teacher_ohe = vectorizer_prefix.transform(X_test['teacher_prefix'].val

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.5 encoding categorical features: project_grade_category

In [26]:

```
vectorizer_grade = LabelEncoderExt()
vectorizer_grade.fit(X_train['project_grade_category'].values)

X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_categor
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.6 encoding numerical features: price

In [27]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X_train['price'].values.reshape(-1,1))
X_train_price=scaler.transform(X_train['price'].values.reshape(-1,1))
X_test_price=scaler.transform(X_test['price'].values.reshape(-1,1))


print(X_train_price.shape)
print(X_test_price.shape)
print(X_train_price)
print(X_test_price)
```

```
(87398, 1)
(21850, 1)
[[0.03464475]
 [0.04108782]
 [0.00068311]
 ...
 [0.04063975]
 [0.01774795]
 [0.02993797]]
[[0.02985896]
 [0.01793598]
 [0.02384096]
 ...
 [0.01587964]
 [0.02398498]
 [0.0349398 ]]
```

1.4.7 encoding numerical features: teacher_number_of_previously_posted_projects

In [28]:

```
scaler = MinMaxScaler()
scaler.fit(X_train['teacher_number_of_previously_posted_projects'].values.res
X_train_pre_proj=scaler.transform(X_train['teacher_number_of_previously_poste
X_test_pre_proj=scaler.transform(X_test['teacher_number_of_previously_posted_


print(X_train_pre_proj.shape)
print(X_test_pre_proj.shape)
```

```
(87398, 1)
(21850, 1)
```

1.4.8 encoding numerical features: quantity

In [29]:

```
scaler = MinMaxScaler()
scaler.fit(X_train['quantity'].values.reshape(-1,1))
X_train_quantity=scaler.transform(X_train['quantity'].values.reshape(-1,1))
X_test_quantity=scaler.transform(X_test['quantity'].values.reshape(-1,1))


print(X_train_quantity.shape)
print(X_test_quantity.shape)
```

```
(87398, 1)
(21850, 1)
```

In [30]:

```
X_train_numerals=np.concatenate((X_train_pre_proj,X_train_quantity,X_train_pr
X_test_numerals=np.concatenate((X_test_pre_proj,X_test_quantity,X_test_price)

print(X_train_numerals.shape)
print(X_test_numerals.shape)
```

```
(87398, 3)
(21850, 3)
```

In [31]:

```
print('vocabulary size is ',vocab_size)
```

```
vocabulary size is  61239
```

In [1]:

```
'''
These cells are to load the data if the system crashes during training
'''
%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.utils import class_weight
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
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
import datetime

import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.initializers import RandomNormal
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Embedding, Dense, Flatten, concate
```

```
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, Tensor
vocab_size = 61239
```

In [32]:

```
#Saving processed data

from sklearn.externals import joblib

joblib.dump(embedding_matrix, 'embedding_matrix2.pkl')
joblib.dump(X_train_padded_docs, 'X_train_padded_docs2.pkl')
joblib.dump(X_train_cc_ohe, 'X_train_cc_ohe2.pkl')
joblib.dump(X_train_csc_ohe, 'X_train_csc_ohe2.pkl')
joblib.dump(X_train_teacher_ohe, 'X_train_teacher_ohe2.pkl')
joblib.dump(X_train_grade_ohe, 'X_train_grade_ohe2.pkl')
joblib.dump(X_train_state_ohe, 'X_train_state_ohe2.pkl')
joblib.dump(X_train_numerals, 'X_train_numerals2.pkl')
joblib.dump(y_train, 'y_train2.pkl')


joblib.dump(X_test_padded_docs, 'X_test_padded_docs2.pkl')
joblib.dump(X_test_cc_ohe, 'X_test_cc_ohe2.pkl')
joblib.dump(X_test_csc_ohe, 'X_test_csc_ohe2.pkl')
joblib.dump(X_test_teacher_ohe, 'X_test_teacher_ohe2.pkl')
joblib.dump(X_test_grade_ohe, 'X_test_grade_ohe2.pkl')
joblib.dump(X_test_state_ohe, 'X_test_state_ohe2.pkl')
joblib.dump(X_test_numerals, 'X_test_numerals2.pkl')
joblib.dump(y_test, 'y_test2.pkl')
```

Out[32]:

```
['y_test2.pkl']
```

In [2]:

```
#Loading processed data

from sklearn.externals import joblib

embedding_matrix = joblib.load('embedding_matrix2.pkl')
X_train_padded_docs = joblib.load('X_train_padded_docs2.pkl')
X_train_cc_ohe = joblib.load('X_train_cc_ohe2.pkl')
X_train_csc_ohe = joblib.load('X_train_csc_ohe2.pkl')
X_train_teacher_ohe = joblib.load('X_train_teacher_ohe2.pkl')
X_train_grade_ohe = joblib.load('X_train_grade_ohe2.pkl')
X_train_state_ohe = joblib.load('X_train_state_ohe2.pkl')
X_train_numerals = joblib.load('X_train_numerals2.pkl')
y_train = joblib.load('y_train2.pkl')


X_test_padded_docs = joblib.load('X_test_padded_docs2.pkl')
X_test_cc_ohe = joblib.load('X_test_cc_ohe2.pkl')
X_test_csc_ohe = joblib.load('X_test_csc_ohe2.pkl')
X_test_teacher_ohe = joblib.load('X_test_teacher_ohe2.pkl')
X_test_grade_ohe = joblib.load('X_test_grade_ohe2.pkl')
X_test_state_ohe = joblib.load('X_test_state_ohe2.pkl')
X_test_numerals = joblib.load('X_test_numerals2.pkl')
y_test = joblib.load('y_test2.pkl')
```

In [3]:

```
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
                        [y_true, y_pred],
                        'float32',
                        stateful=True,
                        name='sklearnAUC' )
    return score
```

Model's architecture

In [4]:

```
# Essay Layers
essay_input = Input(shape=(len(X_train_padded_docs[0]),), name='essay_input')

essay_input_1 = Embedding(input_dim=vocab_size,output_dim=300, input_length=
                          weights=[embedding_matrix], trainable=False)(es

essay_input_1 = CuDNNLSTM(units = 64,
                          kernel_initializer= 'he_normal',
                          return_sequences=True)(essay_input_1)
essay_input_1 = Flatten()(essay_input_1)

# Category Layers
categories_input = Input(shape=(1,), name='categories_input')

categories_input_1= Embedding(input_dim=len(set(X_train_cc_ohe)), output_dim
categories_input_1 = Flatten()(categories_input_1)

# Sub Category Layers
sub_categories_input = Input(shape=(1,), name='sub_categories_input')

sub_categories_input_1 = Embedding(input_dim=len(set(X_train_csc_ohe)), outpu
sub_categories_input_1 = Flatten()(sub_categories_input_1)

# Grade Layers
proj_grade_input = Input(shape=(1,), name='proj_grade_input')

proj_grade_input_1 = Embedding(input_dim=len(set(X_train_grade_ohe)), output_
proj_grade_input_1 = Flatten()(proj_grade_input_1)

# School Layers
school_state_input = Input(shape=(1,), name='school_state_input')

school_state_input_1 = Embedding(input_dim=len(set(X_train_state_ohe)), outpu
school_state_input_1 = Flatten()(school_state_input_1)

# Teacher Prefix Layers
tch_input = Input(shape=(1,), name='tch_input')

tch_input_1= Embedding(input_dim=len(set(X_train_teacher_ohe)), output_dim =
tch_input_1 = Flatten()(tch_input_1)

# Numerical Layers
numeral_input = Input(shape=(X_train_numerals.shape[1],),name='numeral_input'

numeral_input_1 = Dense(units = 64,
                        activation = 'relu',
                        kernel_initializer = 'he_normal')(numeral_input)


# Cconcatinating all the above Layers
```

```
x = concatenate([essay_input_1, categories_input_1, sub_categories_input_1,
                     proj_grade_input_1, school_state_input_1,
                     tch_input_1, numeral_input_1])

output = Dropout(0.5)(x)
output = BatchNormalization()(output)

# Dense Layers
output = Dense(units = 128,
                  activation = 'relu',
                             kernel_initializer = 'he_normal')(output)
output = Dropout(0.8)(output)
output = BatchNormalization()(output)

output = Dense(units = 64,
                  activation = 'relu',
                             kernel_initializer = 'he_normal')(output)

output = Dropout(0.5)(output)
output =Dense(1, activation='sigmoid', name='output')(output)

model_2 = Model(inputs = [essay_input, categories_input, sub_categories_input
                             proj_grade_input, school_state_input, tch_input,
                  outputs=output)

model_2.compile(loss = 'binary_crossentropy',
                  optimizer = 'adam',
                  metrics = ['accuracy', auc])
```

```
WARNING:tensorflow:From c:\users\addu\appdata\local\program
s\python\python37\lib\site-packages\tensorflow\python\keras
\initializers.py:119: calling RandomUniform.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated an
d will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead o
f passing it to the constructor
WARNING:tensorflow:Large dropout rate: 0.8 (>0.5). In Tenso
rFlow 2.x, dropout() uses dropout rate instead of keep_pro
b. Please ensure that this is intended.
WARNING:tensorflow:From c:\users\addu\appdata\local\program
s\python\python37\lib\site-packages\tensorflow\python\ops\i
nit_ops.py:1251: calling VarianceScaling.__init__ (from ten
sorflow.python.ops.init_ops) with dtype is deprecated and w
ill be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead o
f passing it to the constructor
```

Training the model with train data

In [5]:

```
log_dir="logs\\model_2_log" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S
tensorboard = TensorBoard(log_dir=log_dir)

mcp_save = ModelCheckpoint('.model2_best_weights.hdf5', save_best_only=True,

callbacks_list = [mcp_save,tensorboard]
```

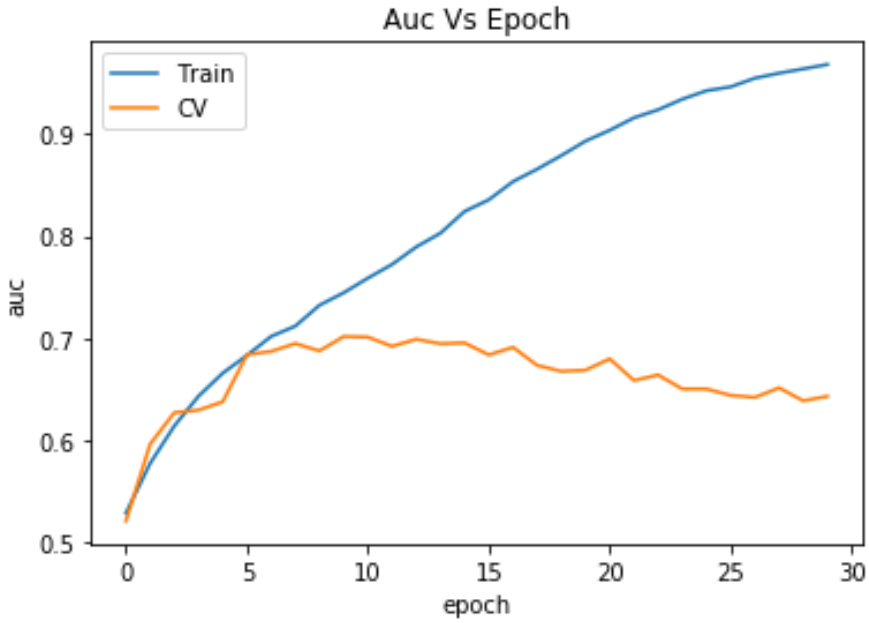
In [6]:

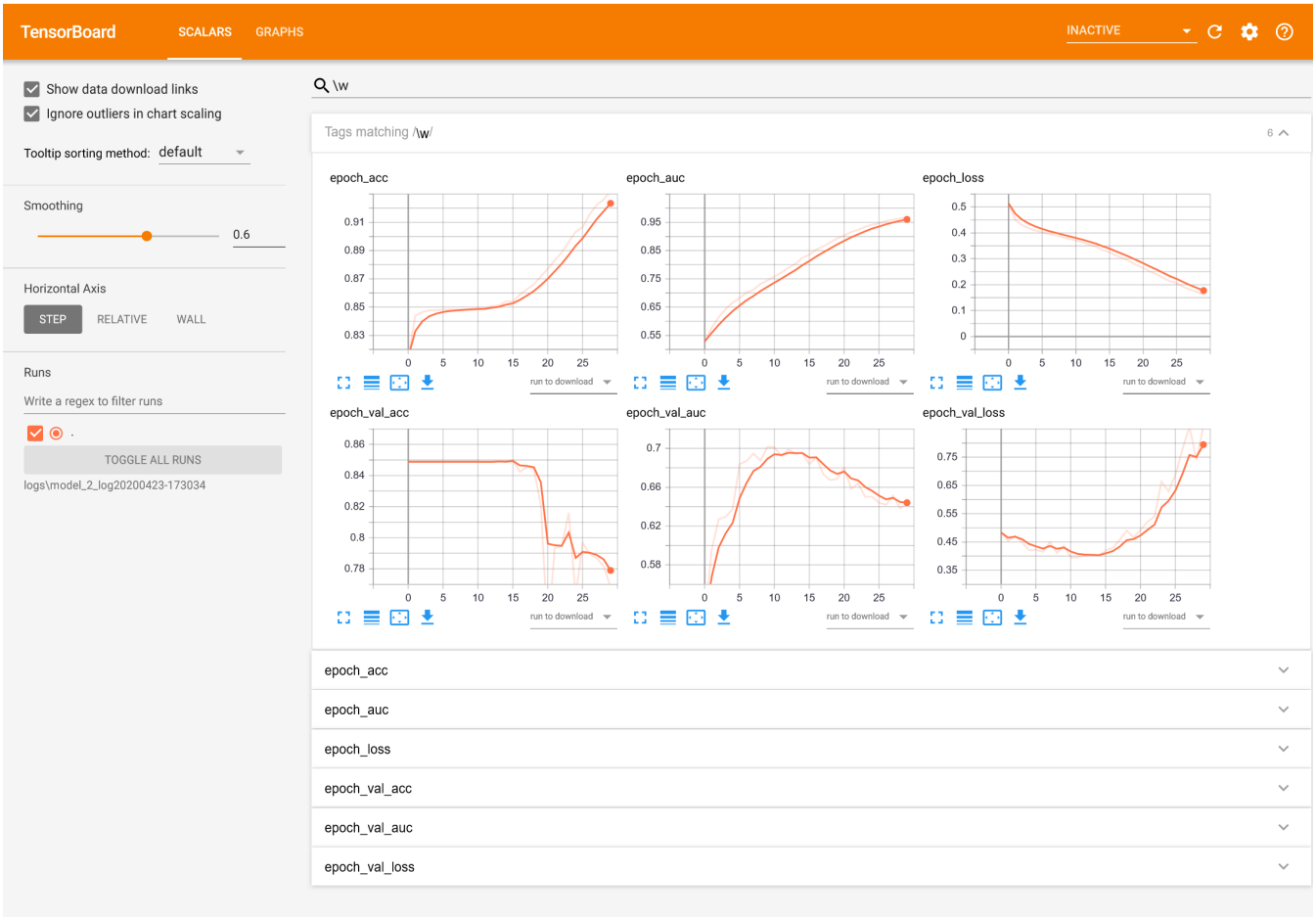
```
history = model_2.fit([X_train_padded_docs, X_train_cc_ohe, X_train_csc_ohe,
                       X_train_grade_ohe, X_train_state_ohe, X_train_teacher_c
                      y_train,
                      batch_size = 512,
                      epochs = 30,
                      callbacks=callbacks_list,
                      validation_split = 0.2,
                      verbose = 2)
```

```
Train on 69918 samples, validate on 17480 samples
Epoch 1/30
69918/69918 - 8s - loss: 0.5140 - acc: 0.8148 - auc: 0.5285
- val_loss: 0.4834 - val_acc: 0.8489 - val_auc: 0.5208
Epoch 2/30
69918/69918 - 6s - loss: 0.4498 - acc: 0.8440 - auc: 0.5772
- val_loss: 0.4540 - val_acc: 0.8489 - val_auc: 0.5961
Epoch 3/30
69918/69918 - 6s - loss: 0.4302 - acc: 0.8465 - auc: 0.6139
- val_loss: 0.4720 - val_acc: 0.8489 - val_auc: 0.6270
Epoch 4/30
69918/69918 - 6s - loss: 0.4182 - acc: 0.8478 - auc: 0.6430
- val_loss: 0.4495 - val_acc: 0.8489 - val_auc: 0.6295
Epoch 5/30
69918/69918 - 6s - loss: 0.4097 - acc: 0.8478 - auc: 0.6656
- val_loss: 0.4208 - val_acc: 0.8489 - val_auc: 0.6376
Epoch 6/30
69918/69918 - 6s - loss: 0.4026 - acc: 0.8482 - auc: 0.6830
- val_loss: 0.4228 - val_acc: 0.8489 - val_auc: 0.6837
Epoch 7/30
```

In [7]:

```
# summarize history for accuracy
plt.plot(history.history['auc'])
plt.plot(history.history['val_auc'])
plt.title('Auc Vs Epoch')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['Train', 'CV'], loc='right bottom')
plt.show()
```





Testing the model with test data

In [8]:

```
# Essay Layers
essay_input = Input(shape=(len(X_train_padded_docs[0]),), name='essay_input')

essay_input_1 = Embedding(input_dim=vocab_size,output_dim=300, input_length=
                          weights=[embedding_matrix], trainable=False)(es

essay_input_1 = CuDNNLSTM(units = 64,
                          kernel_initializer= 'he_normal',
                          return_sequences=True)(essay_input_1)
essay_input_1 = Flatten()(essay_input_1)

# Category Layers
categories_input = Input(shape=(1,), name='categories_input')

categories_input_1= Embedding(input_dim=len(set(X_train_cc_ohe)), output_dim
categories_input_1 = Flatten()(categories_input_1)

# Sub Category Layers
sub_categories_input = Input(shape=(1,), name='sub_categories_input')

sub_categories_input_1 = Embedding(input_dim=len(set(X_train_csc_ohe)), outpu
sub_categories_input_1 = Flatten()(sub_categories_input_1)

# Grade Layers
proj_grade_input = Input(shape=(1,), name='proj_grade_input')

proj_grade_input_1 = Embedding(input_dim=len(set(X_train_grade_ohe)), output_
proj_grade_input_1 = Flatten()(proj_grade_input_1)

# School Layers
school_state_input = Input(shape=(1,), name='school_state_input')

school_state_input_1 = Embedding(input_dim=len(set(X_train_state_ohe)), outpu
school_state_input_1 = Flatten()(school_state_input_1)

# Teacher Prefix Layers
tch_input = Input(shape=(1,), name='tch_input')

tch_input_1= Embedding(input_dim=len(set(X_train_teacher_ohe)), output_dim =
tch_input_1 = Flatten()(tch_input_1)

# Numerical Layers
numeral_input = Input(shape=(X_train_numerals.shape[1],),name='numeral_input'

numeral_input_1 = Dense(units = 64,
                        activation = 'relu',
                        kernel_initializer = 'he_normal')(numeral_input)


# Cconcatinating all the above Layers
```

```
x = concatenate([essay_input_1, categories_input_1, sub_categories_input_1,
                     proj_grade_input_1, school_state_input_1,
                     tch_input_1, numeral_input_1])

output = Dropout(0.5)(x)
output = BatchNormalization()(output)

# Dense Layers
output = Dense(units = 128,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)
output = Dropout(0.8)(output)
output = BatchNormalization()(output)

output = Dense(units = 64,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)

output = Dropout(0.5)(output)
output =Dense(1, activation='sigmoid', name='output')(output)

model_2 = Model(inputs = [essay_input, categories_input, sub_categories_input
                            proj_grade_input, school_state_input, tch_input,n
                  outputs=output)

model_2.compile(loss = 'binary_crossentropy',
                  optimizer = 'adam',
                  metrics = ['accuracy', auc])
```

```
WARNING:tensorflow:Large dropout rate: 0.8 (>0.5). In TensorFl
ow 2.x, dropout() uses dropout rate instead of keep_prob. Plea
se ensure that this is intended.
```

In [9]:

```
#Loading the best weights obtained from traing model on train data
model_2.load_weights(".model2_best_weights.hdf5")
```

In [10]:

```
score=model_2.evaluate([X_test_padded_docs, X_test_cc_ohe, X_test_csc_ohe,
                                  X_test_grade_ohe, X_test_state_ohe, X_test_t
                           batch_size=512, verbose=0)[2]
```

Auc score of model on unseen Test data

In [11]:

```
print('Test Auc obtained is ',score)
```

```
Test Auc obtained is  0.70389456
```

In [12]:

```
model_2.summary()
```

```
Model: "model_1"
__________________________________________________________________________
_____________________________________
Layer (type)                    Output Shape         Param
#     Connected to
==========================================================
=====================================
essay_input (InputLayer)        [(None, 300)]        0
__________________________________________________________________________
_____________________________________
embedding_6 (Embedding)         (None, 300, 300)     183717
00    essay_input[0][0]
__________________________________________________________________________
_____________________________________
categories_input (InputLayer)   [(None, 1)]          0
__________________________________________________________________________
_____________________________________
sub_categories_input (InputLaye [(None, 1)]          0
__________________________________________________________________________
_____________________________________
proj_grade_input (InputLayer)   [(None, 1)]          0
__________________________________________________________________________
_____________________________________
school_state_input (InputLayer) [(None, 1)]          0
__________________________________________________________________________
_____________________________________
tch_input (InputLayer)          [(None, 1)]          0
__________________________________________________________________________
_____________________________________
cu_dnnlstm_1 (CuDNNLSTM)        (None, 300, 64)      93696
embedding_6[0][0]
__________________________________________________________________________
_____________________________________
embedding_7 (Embedding)         (None, 1, 16)        816
categories_input[0][0]
__________________________________________________________________________
_____________________________________
embedding_8 (Embedding)         (None, 1, 64)        25216
sub_categories_input[0][0]
__________________________________________________________________________
_____________________________________
embedding_9 (Embedding)         (None, 1, 64)        256
proj_grade_input[0][0]
__________________________________________________________________________
_____________________________________
embedding_10 (Embedding)        (None, 1, 64)        3264
school_state_input[0][0]
__________________________________________________________________________
```

```
__________________________________________________
embedding_11 (Embedding)        (None, 1, 64)        320
tch_input[0][0]
__________________________________________________
__________________________________________________
numeral_input (InputLayer)      [(None, 3)]          0
__________________________________________________
__________________________________________________
flatten_6 (Flatten)             (None, 19200)        0
cu_dnnlstm_1[0][0]
__________________________________________________
__________________________________________________
flatten_7 (Flatten)             (None, 16)           0
embedding_7[0][0]
__________________________________________________
__________________________________________________
flatten_8 (Flatten)             (None, 64)           0
embedding_8[0][0]
__________________________________________________
__________________________________________________
flatten_9 (Flatten)             (None, 64)           0
embedding_9[0][0]
__________________________________________________
__________________________________________________
flatten_10 (Flatten)            (None, 64)           0
embedding_10[0][0]
__________________________________________________
__________________________________________________
flatten_11 (Flatten)            (None, 64)           0
embedding_11[0][0]
__________________________________________________
__________________________________________________
dense_3 (Dense)                 (None, 64)           256
numeral_input[0][0]
__________________________________________________
__________________________________________________
concatenate_1 (Concatenate)     (None, 19536)        0
flatten_6[0][0]

flatten_7[0][0]

flatten_8[0][0]

flatten_9[0][0]

flatten_10[0][0]

flatten_11[0][0]

dense_3[0][0]
__________________________________________________
__________________________________________________
dropout_3 (Dropout)             (None, 19536)        0
```

```
concatenate_1[0][0]
__________________________________________________________________
_____________________________________
batch_normalization_2 (BatchNor (None, 19536)        78144
dropout_3[0][0]
__________________________________________________________________
_____________________________________
dense_4 (Dense)                 (None, 128)          250073
6     batch_normalization_2[0][0]
__________________________________________________________________
_____________________________________
dropout_4 (Dropout)             (None, 128)          0
dense_4[0][0]
__________________________________________________________________
_____________________________________
batch_normalization_3 (BatchNor (None, 128)          512
dropout_4[0][0]
__________________________________________________________________
_____________________________________
dense_5 (Dense)                 (None, 64)           8256
batch_normalization_3[0][0]
__________________________________________________________________
_____________________________________
dropout_5 (Dropout)             (None, 64)           0
dense_5[0][0]
__________________________________________________________________
_____________________________________
output (Dense)                  (None, 1)            65
dropout_5[0][0]
==================================================================
====================================
Total params: 21,083,237
Trainable params: 2,672,209
Non-trainable params: 18,411,028
__________________________________________________________________
_____________________________________
```

Model's Final Architecture

In [13]:

```
#drawing models
tf.keras.utils.plot_model(
    model_2,
    show_shapes=False,
    show_layer_names=True,
    rankdir='TB'
)
```

Out[13]:

Summary

- Created tensorflow model using text, categorical and numerical layers.
- Removed the words from the text whos idf is either very small or very large
- Performed hyper parameter tuning manually on number of layers, activation functions and optimizers.

- Trained the network using the best obtained hyper parameters.
- Tested the model using Test data and obtained test aucroc score of 0.70.
- Printed the summary of the model along with it's image.