Assignment : DT

Please check below video before attempting this assignment

In [1]:

```
from IPython.display import YouTubeVideo
YouTubeVideo('ZhLXULFjIjQ', width="1000",height="500")
```

Out[1]:

TF-IDFW2V

Tfidf w2v (w1,w2..) = (tfidf(w1) * w2v(w1) + tfidf(w2) * w2v(w2) + ...) / (tfidf(w1) + tfidf(w2) + ...)

(Optional) Please check course video on AVgw2V and TF-IDFW2V for more details.

Glove vectors

In this assignment you will be working with glove vectors , please check [this] (https://en.wikipedia.org/wiki/GloVe_(machine_learning)) and [this] (https://en.wikipedia.org/wiki/GloVe_(machine_learning)) for more details.

Download glove vectors from this link

In [2]:

```
'''#please use below code to load glove vectors
with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/glove_vectors', 'rb') as f:
```

```
    model = pickle.load(f)
    glove_words =  set(model.keys())'''
```

Out[2]:

```
"#please use below code to load glove vectors \nwith open('/content/drive/MyDrive/9_Donors_choose_DT/Data/glove_vectors', 'rb') as f:\n    model = pickle.load(f)\n    glove_words =  set(model.keys())"
```

or else , you can use below code

In [3]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
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')

# ==============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ==============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)


'''
```

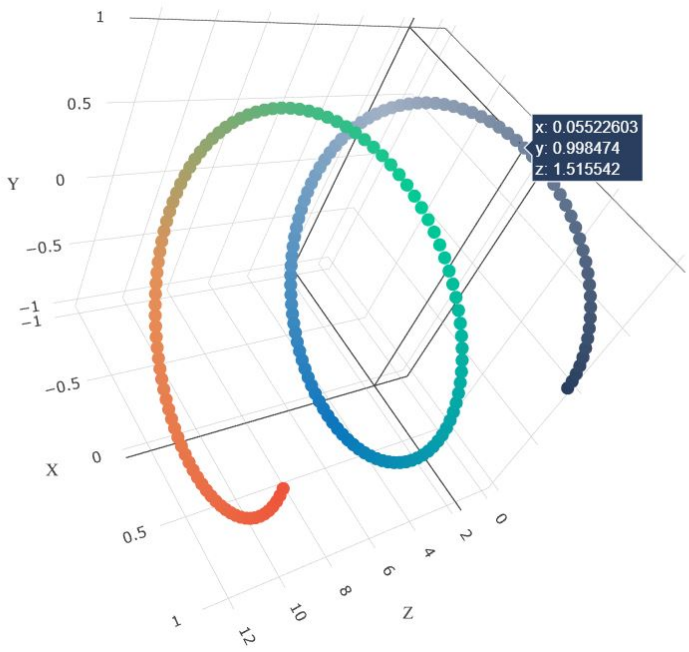
Out[3]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef l
```

```
oadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\
', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.s
plit()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in
splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words
loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ========
====================\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\
nDone. 1917495  words loaded!\n\n# ============================\n\nwords = []\nfor i in p
reproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    word
s.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(w
ords)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.key
s()).intersection(words)\nprint("The number of words that are present in both glove vecto
rs and our coupus",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3
),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i
in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(word
s_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayung.c
om/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'
glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

Task - 1

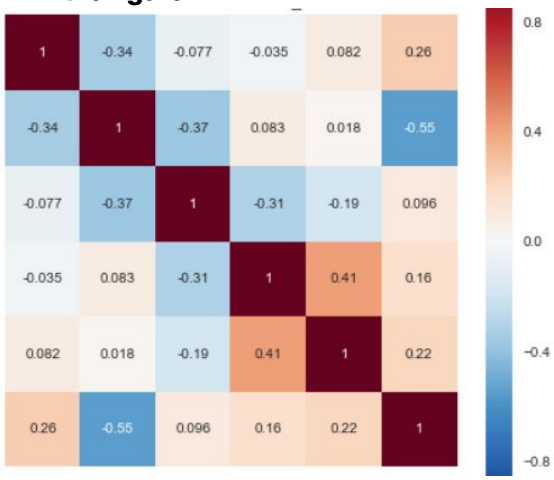
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - **Set 1: categorical, numerical features + preprocessed_essay (TFIDF) + Sentiment scores(preprocessed_essay)**
    - **Set 2: categorical, numerical features + preprocessed_essay (TFIDF W2V) + Sentiment scores(preprocessed_essay)**
      ** **
    - **The hyper paramter tuning (best `depth` in range [1, 3, 10, 30], and the best `min_samples_split` in range [5, 10, 100, 500])**
        - **Find the best hyper parameter which will give the maximum AUC value**
        - **find the best hyper paramter using k-fold cross validation(use gridsearch cv or randomsearch cv)/simple cross validation data(you can write your own for loops refer sample solution)**
          ** **
        - **Representation of results**
            - **You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure**



with X-axis as min_sample_split, Y-axis as max_depth, and Z-axis as AUC Score , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

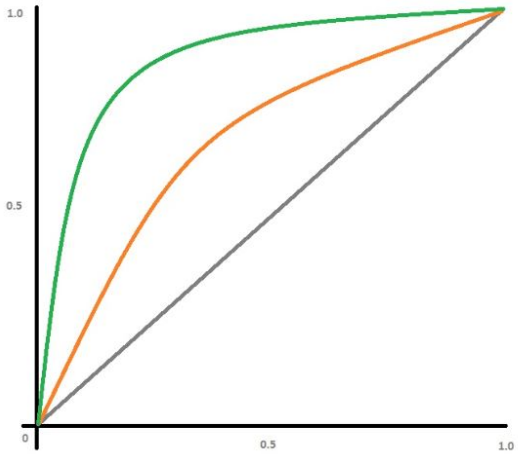
or

- **You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure**



[seaborn heat maps] with rows as min_sample_split, columns as max_depth, and values inside the cell representing AUC Score

- **You choose either of the plotting techniques out of 3d plot or heat map**
- **Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test. Make sure that you are using predict_proba method to calculate AUC curves, because AUC is calcualted on class probabilities and not on class labels.**



- **Along with plotting ROC curve, you need to print the [confusion matrix] with predicted and original labels of test data points**

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

- **Once after you plot the confusion matrix with the test data, get all the `false positive data points`**
  - **Plot the WordCloud(https://www.geeksforgeeks.org/generating-word-cloud-python/) with the words of essay text of these `false positive data points`**
  - **Plot the box plot with the `price` of these `false positive data points`**
  - **Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`**

** **

In [3]:

Task - 2

For this task consider set-1 features.

- **Select all the features which are having non-zero feature importance.You can get the feature importance using 'feature*importances*` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html), discard the all other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM).**
- **You need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3**
  **Note: when you want to find the feature importance make sure you don't use max_depth parameter keep it None. **
  You need to summarize the results at the end of the notebook, summarize it in the table format

  <img src='http://i.imgur.com/YVpIGGE.jpg' width=400px>

  ** **

Hint for calculating Sentiment scores

In [4]:

```
# import nltk
# nltk.download('vader_lexicon')
```

In [5]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# import nltk
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

sample_sentence_1='I am happy.'
ss_1 = sid.polarity_scores(sample_sentence_1)
print('sentiment score for sentence 1',ss_1)

sample_sentence_2='I am sad.'
ss_2 = sid.polarity_scores(sample_sentence_2)
print('sentiment score for sentence 2',ss_2)

sample_sentence_3='I am going to New Delhi tommorow.'
ss_3 = sid.polarity_scores(sample_sentence_3)
print('sentiment score for sentence 3',ss_3)
```

```
sentiment score for sentence 1 {'neg': 0.0, 'neu': 0.213, 'pos': 0.787, 'compound': 0.571
9}
sentiment score for sentence 2 {'neg': 0.756, 'neu': 0.244, 'pos': 0.0, 'compound': -0.47
67}
sentiment score for sentence 3 {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Decision Tree

Task - 1

In [6]:

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

```
Mounted at /content/drive
```

import section

In [172]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,roc_auc_score,confusion_matrix,classification_
report,f1_score,log_loss,roc_curve,plot_roc_curve
from tqdm import tqdm
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
import warnings
warnings.filterwarnings('ignore')
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
#!pip install scikit-plot
#import scikitplot as skplt
from wordcloud import WordCloud, STOPWORDS
import os
```

Output hidden; open in https://colab.research.google.com to view.

Creating the results table

In [173]:

```
result_table = pd.DataFrame(data = np.zeros((4,5)),columns = ['Vectorizer','Model','Hype
r_Param_Depth','Hyper_Param_Min_Sample_Split','AUC'])
result_table['Vectorizer'][0] = 'TFIDF'
result_table['Vectorizer'][1] = 'TFIDFW2V'
result_table['Vectorizer'][2] =  'TFIDF'
result_table['Vectorizer'][3] = 'TFIDFW2V'
result_table['Model'][0] = 'BRUTE'
result_table['Model'][1] = 'BRUTE'
result_table['Model'][2] =  'Non_Zero_Features_Removed'
result_table['Model'][3] = 'Non_Zero_Features_Removed'
result_table
```

Out[173]:

	Vectorizer	Model	Hyper_Param_Depth	Hyper_Param_Min_Sample_Split	AUC
0	TFIDF	BRUTE	0.0	0.0	0.0
1	TFIDFW2V	BRUTE	0.0	0.0	0.0
2	TFIDF	Non_Zero_Features_Removed	0.0	0.0	0.0
3	TFIDFW2V	Non_Zero_Features_Removed	0.0	0.0	0.0

1.1 Loading Data

In [174]:

```
#make sure you are loading atleast 50k datapoints
#you can wo rk with features of preprocessed_data.csv for the assignment.
import pandas
data = pandas.read_csv('/content/drive/MyDrive/9_Donors_choose_DT/preprocessed_data.csv',
```

```
nrows=50000)
```

In [175]:

```
data.shape
```

Out[175]:

```
(50000, 9)
```

In [176]:

```
# write your code in following steps for task 1
# 1. calculate sentiment scores for the essay feature
# 2. Split your data.
# 3. perform tfidf vectorization of text data.
# 4. perform tfidf w2v vectorization of text data.
# 5. perform encoding of categorical features.
# 6. perform encoding of numerical features
# 7. For task 1 set 1 stack up all the features
# 8. For task 1 set 2 stack up all the features (for stacking dense features you can use
np.stack)
# 9. Perform hyperparameter tuning and plot either heatmap or 3d plot.
# 10. Find the best parameters and fit the model. Plot ROC-AUC curve(using predict proba
method)
# 11. Plot confusion matrix based on best threshold value
# 12. Find all the false positive data points and plot wordcloud of essay text and pdf of
teacher_number_of_previously_posted_projects.
# 13. Write your observations about the wordcloud and pdf.
```

In [177]:

```
# please write all the code with proper documentation, and proper titles for each subsect
ion
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your c
ode
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Step 1 : calculate sentiment scores for the essay feature

In [178]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#nltk.download('vader_lexicon')
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/data_step1.pkl'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/data_step1.pkl', 'rb') as fil
e:
    data = pickle.load(file)
else:
  sid = SentimentIntensityAnalyzer()
  neg = []
  neu = []
  pos = []
  compound = []
  for essay in tqdm(data.essay):
    ss_1 = sid.polarity_scores(essay)
    neg.append(ss_1['neg'])
    neu.append(ss_1['neu'])
    pos.append(ss_1['pos'])
    compound.append(ss_1['compound'])
  data['essay_neg_sentiment'] = neg
  data['essay_neu_sentiment'] = neu
```

```
  data['essay_pos_sentiment'] = pos
  data['essay_comp_sentiment'] = compound
```

Save step1

In [179]:

```
if not os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/data_step1.pkl'):
  with open('data_step1.pkl', 'wb') as file:
      pickle.dump(data, file)
  !cp data_step1.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

In [180]:

```
data.head(1)
```

Out[180]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	cl
0	ca	mrs	grades_prek_2	53	1	

Step 2. Split your data

In [181]:

```
X = data.drop(['project_is_approved'],axis = 1)
y = data[['project_is_approved']]
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state = 2,stratify = y,test_
size=0.2)
X_train = X_train.reset_index(drop=True)
X_test = X_test.reset_index(drop=True)
y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(40000, 12)
(10000, 12)
(40000, 1)
(10000, 1)
```

In [182]:

```
X_train_sentiment = np.array(X_train[['essay_neg_sentiment','essay_neu_sentiment','essay
_pos_sentiment','essay_comp_sentiment']])
X_test_sentiment = np.array(X_test[['essay_neg_sentiment','essay_neu_sentiment','essay_p
os_sentiment','essay_comp_sentiment']])
print(X_train_sentiment.shape)
print(X_test_sentiment.shape)
```

```
(40000, 4)
(10000, 4)
```

Step 3. perform tfidf vectorization of text data.

In [183]:

```
vectorizer_essay_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features = 7000
)
vectorizer_essay_tfidf.fit(X_train['essay'].values) # fit has to happen only on train dat
a
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_train_essay_tfidf.pkl
'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_train_essay_tfidf.pkl', 'rb
') as file:
    X_train_essay_tfidf = pickle.load(file)
  with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_test_essay_tfidf.pkl', 'rb'
) as file:
    X_test_essay_tfidf = pickle.load(file)
else:
  # we use the fitted CountVectorizer to convert the text to vector
  X_train_essay_tfidf = vectorizer_essay_tfidf.transform(X_train['essay'].values)
  X_test_essay_tfidf = vectorizer_essay_tfidf.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(40000, 7000) (40000, 1)
(10000, 7000) (10000, 1)
====================================================================================================
```

In [184]:

```
X_train_essay_tfidf_columms = vectorizer_essay_tfidf.get_feature_names()
```

In [185]:

```
with open('X_train_essay_tfidf.pkl', 'wb') as file:
    pickle.dump(X_train_essay_tfidf, file)
!cp X_train_essay_tfidf.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data

with open('X_test_essay_tfidf.pkl', 'wb') as file:
    pickle.dump(X_test_essay_tfidf, file)
!cp X_test_essay_tfidf.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

Step 4. perform tfidf w2v vectorization of text data.

In [186]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pic
kle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('/content/drive/MyDrive/9_Donors_choose_DT/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [187]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [188]:

```
# average Word2Vec
# compute average word2vec for each review.
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_train_essay_tfidf_w2v
.pkl'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_train_essay_tfidf_w2v.pkl',
'rb') as file:
```

```
    X_train_essay_tfidf_w2v = pickle.load(file)
else:
  X_train_essay_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this
list
  for sentence in tqdm(X_train['essay'].values): # for each review/sentence
      vector = np.zeros(300) # as word vectors are of zero length
      tf_idf_weight =0; # num of words with a valid vector in the sentence/review
      for word in sentence.split(): # for each word in a review/sentence
          if (word in glove_words) and (word in tfidf_words):
              vec = model[word] # getting the vector for each word
              # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
              tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # g
etting the tfidf value for each word
              vector += (vec * tf_idf) # calculating tfidf weighted w2v
              tf_idf_weight += tf_idf
      if tf_idf_weight != 0:
          vector /= tf_idf_weight
      X_train_essay_tfidf_w2v.append(vector)
```

In [189]:

```
X_train_essay_tfidf_w2v = np.array(X_train_essay_tfidf_w2v)
X_train_essay_tfidf_w2v.shape
```

Out[189]:

```
(40000, 300)
```

In [190]:

```
with open('X_train_essay_tfidf_w2v.pkl', 'wb') as file:
    pickle.dump(X_train_essay_tfidf_w2v, file)
!cp X_train_essay_tfidf_w2v.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

In [191]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_test_essay_tfidf_w2v.
pkl'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/Data/X_test_essay_tfidf_w2v.pkl',
'rb') as file:
    X_test_essay_tfidf_w2v = pickle.load(file)
else:
  # average Word2Vec
  # compute average word2vec for each review.
  X_test_essay_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this l
ist
  for sentence in tqdm(X_test['essay'].values): # for each review/sentence
      vector = np.zeros(300) # as word vectors are of zero length
      tf_idf_weight =0; # num of words with a valid vector in the sentence/review
      for word in sentence.split(): # for each word in a review/sentence
          if (word in glove_words) and (word in tfidf_words):
              vec = model[word] # getting the vector for each word
              # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
              tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # g
etting the tfidf value for each word
              vector += (vec * tf_idf) # calculating tfidf weighted w2v
              tf_idf_weight += tf_idf
      if tf_idf_weight != 0:
          vector /= tf_idf_weight
      X_test_essay_tfidf_w2v.append(vector)
```

In [192]:

```
X_test_essay_tfidf_w2v = np.array(X_test_essay_tfidf_w2v)
X_test_essay_tfidf_w2v.shape
```

Out[192]:

```
(10000, 300)
```

In [193]:

```
with open('X_test_essay_tfidf_w2v.pkl', 'wb') as file:
    pickle.dump(X_test_essay_tfidf_w2v, file)
!cp X_test_essay_tfidf_w2v.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

Step 5. perform encoding of categorical features.

1 encoding categorical features: School State

In [194]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 51) (40000, 1)
(10000, 51) (10000, 1)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il'
, 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne
', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'u
t', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
====================================================================================
==========
```

In [195]:

```
categorical_features = vectorizer.get_feature_names()
```

2 encoding categorical features: teacher_prefix

In [196]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 5) (40000, 1)
(10000, 5) (10000, 1)
['dr', 'mr', 'mrs', 'ms', 'teacher']
====================================================================================
==========
```

In [197]:

```
categorical_features += vectorizer.get_feature_names()
```

3 encoding categorical features: project_grade_category

In [198]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only on trai
n data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 4) (40000, 1)
(10000, 4) (10000, 1)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================
===========
```

In [199]:

```
categorical_features += vectorizer.get_feature_names()
```

4 encoding categorical features: clean_categories

In [200]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_categories_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_clean_categories_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_categories_ohe.shape, y_train.shape)
print(X_test_clean_categories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 9) (40000, 1)
(10000, 9) (10000, 1)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language'
, 'math_science', 'music_arts', 'specialneeds', 'warmth']
====================================================================================
===========
```

In [201]:

```
categorical_features += vectorizer.get_feature_names()
```

5 encoding categorical features: clean_subcategories

In [202]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train d
ata

# we use the fitted CountVectorizer to convert the text to vector
```

```
X_train_clean_subcategories_ohe = vectorizer.transform(X_train['clean_subcategories'].val
ues)
X_test_clean_subcategories_ohe = vectorizer.transform(X_test['clean_subcategories'].value
s)

print("After vectorizations")
print(X_train_clean_subcategories_ohe.shape, y_train.shape)
print(X_test_clean_subcategories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 30) (40000, 1)
(10000, 30) (10000, 1)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_ca
reerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience', '
esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness', 'health_
lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', '
mathematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingart
s', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
====================================================================================
==========
```

In [203]:

```
categorical_features += vectorizer.get_feature_names()
```

Step 6. perform encoding of numerical features

1 encoding numerical features: Price

In [204]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(40000, 1) (40000, 1)
(10000, 1) (10000, 1)
====================================================================================
==========
```

In [205]:

```
numerical_features = ['price','teacher_number_of_previously_posted_projects']
```

2 encoding numerical features: teacher_number_of_previously_posted_projects

In [206]:

```
normalizer = Normalizer()
# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1
,1))

X_train_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_train[
'teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_test['t
eacher_number_of_previously_posted_projects'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, y_train.shape)
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(40000, 1) (40000, 1)
(10000, 1) (10000, 1)
====================================================================================================
```

In [206]:

Step 7. For task 1 set 1 stack up all the features

In [207]:

```
X_train_essay_tfidf.shape
```

Out[207]:

```
(40000, 7000)
```

In [208]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set1 = hstack((X_train_sentiment,X_train_essay_tfidf, X_train_state_ohe, X_train_te
acher_ohe, X_train_grade_ohe,X_train_clean_categories_ohe,X_train_clean_subcategories_ohe
, X_train_price_norm,X_train_teacher_number_of_previously_posted_projects_norm)).tocsr()
X_te_set1 = hstack((X_test_sentiment,X_test_essay_tfidf, X_test_state_ohe, X_test_teacher
_ohe, X_test_grade_ohe,X_test_clean_categories_ohe,X_test_clean_subcategories_ohe, X_test
_price_norm,X_test_teacher_number_of_previously_posted_projects_norm)).tocsr()

print("Final Data matrix")
print(X_tr_set1.shape, y_train.shape)
print(X_te_set1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(40000, 7105) (40000, 1)
(10000, 7105) (10000, 1)
====================================================================================================
```

Save set1

In [209]:

```
with open('set1_train.pkl', 'wb') as file:
```

```
    pickle.dump(X_tr_set1, file)
!cp set1_train.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
with open('set1_test.pkl', 'wb') as file:
  pickle.dump(X_te_set1, file)
!cp set1_test.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

Load set1

In [210]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/set1_train.pkl'):
  with open('set1_train.pkl', 'rb') as file:
    X_tr_set1 = pickle.load(file)
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/set1_test.pkl'):
  with open('set1_test.pkl', 'rb') as file:
    X_te_set1 = pickle.load(file)
```

Step 8. For task 1 set 2 stack up all the features

In [211]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set2 = hstack((X_train_sentiment,X_train_essay_tfidf_w2v, X_train_state_ohe, X_trai
n_teacher_ohe, X_train_grade_ohe,X_train_clean_categories_ohe,X_train_clean_subcategories
_ohe, X_train_price_norm,X_train_teacher_number_of_previously_posted_projects_norm)).tocs
r()
X_te_set2 = hstack((X_test_sentiment,X_test_essay_tfidf_w2v, X_test_state_ohe, X_test_te
acher_ohe, X_test_grade_ohe,X_test_clean_categories_ohe,X_test_clean_subcategories_ohe, X
_test_price_norm,X_test_teacher_number_of_previously_posted_projects_norm)).tocsr()

print("Final Data matrix")
print(X_tr_set2.shape, y_train.shape)
print(X_te_set2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(40000, 405) (40000, 1)
(10000, 405) (10000, 1)
====================================================================================
==========
```

Save set2

In [212]:

```
with open('set2_train.pkl', 'wb') as file:
    pickle.dump(X_tr_set2, file)
!cp set2_train.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
with open('set2_test.pkl', 'wb') as file:
  pickle.dump(X_te_set2, file)
!cp set2_test.pkl /content/drive/MyDrive/9_Donors_choose_DT/Data
```

Load set2

In [213]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/set2_train.pkl'):
  with open('set2_train.pkl', 'rb') as file:
    X_tr_set2 = pickle.load(file)
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/Data/set2_test.pkl'):
  with open('set2_test.pkl', 'rb') as file:
    X_te_set2 = pickle.load(file)
```

Step 9. Perform hyperparameter tuning and plot either heatmap or 3d plot.

The hyper paramter tuning (best `depth` in range [1, 3, 10, 30], and the best `min_samples_split` in range [5, 10, 100, 500]

Set 1

In [214]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_grid.pkl'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_grid.pkl', 'rb') as fi
le:
    clf = pickle.load(file)
else:
  classif = DecisionTreeClassifier(random_state = 42)
  parameters = {'max_depth': [1, 3, 10, 30],'min_samples_split': [5, 10, 100, 500]}
  clf = GridSearchCV(classif, parameters, cv=5, scoring='roc_auc',return_train_score=Tru
e)
  clf.fit(X_tr_set1, y_train)
```

In [215]:

```
if not os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_grid.pkl'):
  with open('set1_grid.pkl', 'rb') as file:
    pickle.dump(clf, file)
  !cp set1_grid.pkl /content/drive/MyDrive/9_Donors_choose_DT/models
```

In [216]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results.head(5)
```

Out[216]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	par
0	0.440606	0.040823	0.006886	0.001332	1	5	{'max_depth 'min_samples_s
1	0.425683	0.007814	0.004982	0.000252	1	10	{'max_depth 'min_samples_s
2	0.421047	0.000968	0.004793	0.000043	1	100	{'max_depth 'min_samples_s
3	0.422200	0.002031	0.005237	0.000535	1	500	{'max_depth 'min_samples_s
4	1.192830	0.002929	0.005555	0.001230	3	5	{'max_depth 'min_samples_s

5 rows × 22 columns

In [217]:

```
result_required = results[['param_max_depth', 'param_min_samples_split','mean_test_score
','mean_train_score']]
result_required.head(5)
```

Out[217]:

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score
0	1	5	0.548110	0.550340
1	1	10	0.548110	0.550340
2	1	100	0.548110	0.550340
3	1	500	0.548110	0.550340
4	3	5	0.584834	0.594891

3D scatter plot

In [218]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=result_required['param_max_depth'],y=result_required['param_min_
samples_split'],z=result_required['mean_train_score'], name = 'train')
trace2 = go.Scatter3d(x=result_required['param_max_depth'],y=result_required['param_min_
samples_split'],z=result_required['mean_test_score'], name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(width=1000,
    height=1000,scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
fig.show(renderer = 'colab')
plt.show()
```

Heat Map

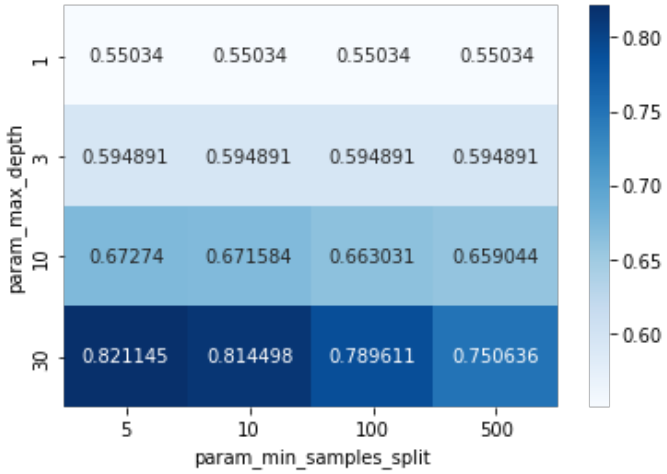
Heat Map For Train

In [219]:

```
sns.heatmap(result_required.pivot('param_max_depth', 'param_min_samples_split', 'mean_tra
in_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[219]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4595f4b610>
```



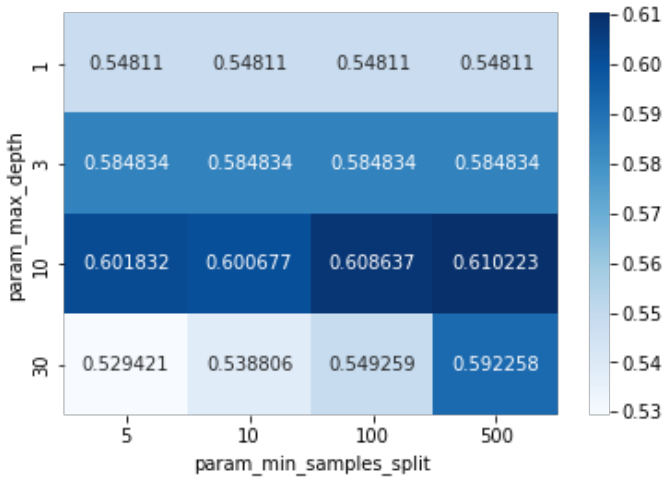
Heat Map For Test

In [220]:

```
sns.heatmap(result_required.pivot('param_max_depth', 'param_min_samples_split', 'mean_tes
t_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[220]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f45966467d0>
```



Set2

In [221]:

```
y_train.shape
```

Out[221]:

```
(40000, 1)
```

In [222]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_grid.pkl'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_grid.pkl', 'rb') as fi
le:
    clf_set2 = pickle.load(file)
else:
  classif_set2 = DecisionTreeClassifier(random_state = 42)
  parameters = {'max_depth': [1, 3, 10, 30],'min_samples_split': [5, 10, 100, 500]}
  clf_set2 = GridSearchCV(classif_set2, parameters, cv=5, scoring='roc_auc',return_train
_score=True)
  clf_set2.fit(X_tr_set2, y_train)
```

In [223]:

```
if not os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_grid.pkl'):
  with open('set2_grid.pkl', 'wb') as file:
      pickle.dump(clf_set2, file)
  !cp set2_grid.pkl /content/drive/MyDrive/9_Donors_choose_DT/models
```

In [224]:

```
results = pd.DataFrame.from_dict(clf_set2.cv_results_)
results.head(5)
```

Out[224]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	par
0	2.272133	0.466181	0.026507	0.002641	1	5	{'max_depth 'min_samples_s
1	2.366318	0.337327	0.028667	0.003538	1	10	{'max_depth 'min_samples_s
2	2.576887	0.290536	0.029645	0.004224	1	100	{'max_depth 'min_samples_s
3	2.549123	0.613357	0.025457	0.004365	1	500	{'max_depth 'min_samples_s
4	6.160511	0.527142	0.026420	0.003631	3	5	{'max_depth 'min_samples_s

5 rows × 22 columns

In [225]:

```
result_required_set2 = results[['param_max_depth', 'param_min_samples_split','mean_test_
score','mean_train_score']]
result_required_set2.head(5)
```

Out[225]:

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score
0	1	5	0.545406	0.556911
1	1	10	0.545406	0.556911
2	1	100	0.545406	0.556911
3	1	500	0.545406	0.556911
4	3	5	0.590709	0.615006

3D scatter plot

In [226]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=result_required_set2['param_max_depth'],y=result_required_set2['
param_min_samples_split'],z=result_required_set2['mean_train_score'], name = 'train')
trace2 = go.Scatter3d(x=result_required_set2['param_max_depth'],y=result_required_set2['
param_min_samples_split'],z=result_required_set2['mean_test_score'], name = 'Cross valid
ation')
data = [trace1, trace2]

layout = go.Layout(width=1000,
    height=1000,scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
fig.show(renderer = 'colab')
plt.show()
```

Heat Map

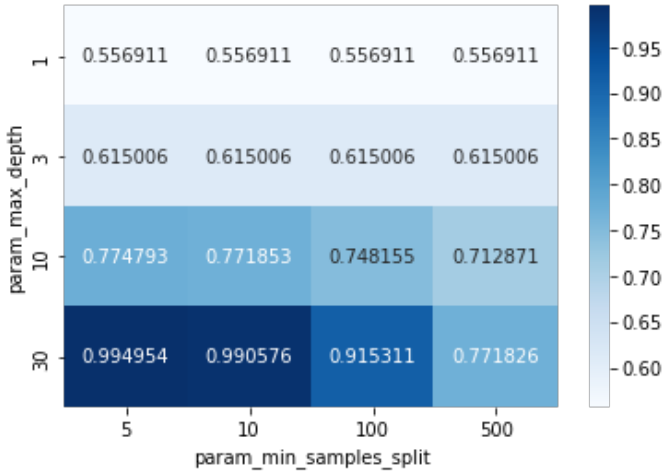
Heat Map For Train

In [227]:

```
sns.heatmap(result_required_set2.pivot('param_max_depth', 'param_min_samples_split', 'mean_train_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[227]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459c0ba3d0>
```



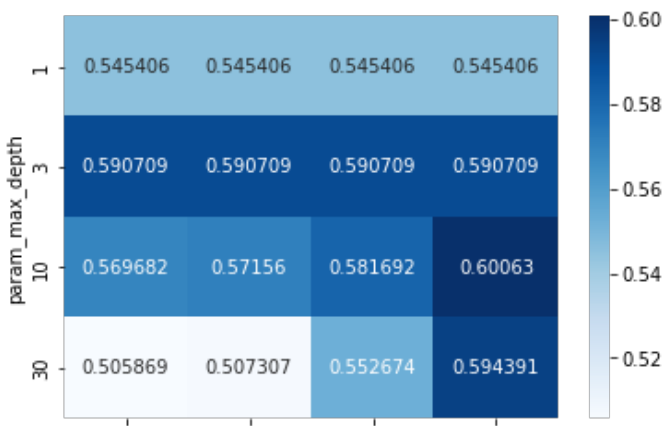
Heat Map For Test

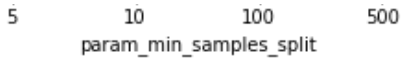
In [228]:

```
sns.heatmap(result_required_set2.pivot('param_max_depth', 'param_min_samples_split', 'mean_test_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[228]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4597e47f90>
```





In [228]:

```
```

Step 10. Find the best parameters and fit the model. Plot ROC-AUC curve(using predict proba method)

SET1

In [229]:

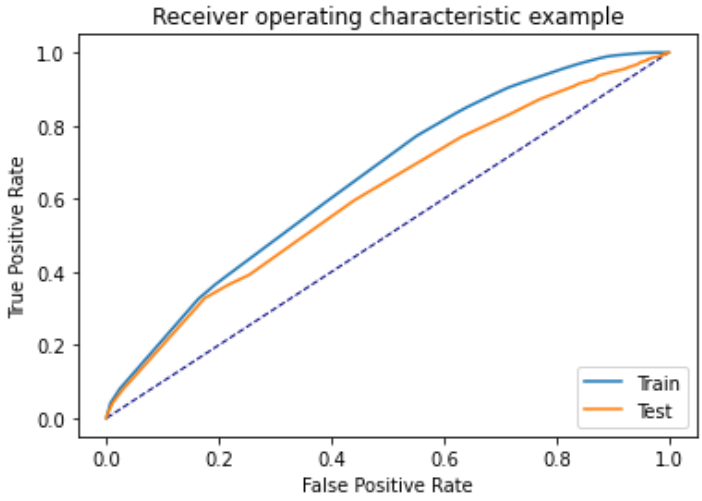
```
classif_set1 = DecisionTreeClassifier(max_depth = clf.best_estimator_.max_depth , min_sa
mples_split = clf.best_estimator_.min_samples_split , random_state = clf.best_estimator_
.random_state)
classif_set1.fit(X_tr_set1, y_train)
```

Out[229]:

```
DecisionTreeClassifier(max_depth=10, min_samples_split=500, random_state=42)
```

In [230]:

```
y_train_pred_set1 = classif_set1.predict(X_tr_set1)
y_test_pred_set1 = classif_set1.predict(X_te_set1)
y_train_pred_proba_set1 = classif_set1.predict_proba(X_tr_set1)[:,[1]]
y_test_pred_proba_set1 = classif_set1.predict_proba(X_te_set1)[:,[1]]
fpr_train_set1, tpr_train_set1, thresholds_train_set1 = roc_curve(y_train,y_train_pred_pr
oba_set1)
fpr_test_set1, tpr_test_set1, thresholds_test_set1 = roc_curve(y_test,y_test_pred_proba_s
et1)
plt.plot([0, 1], [0, 1], color="navy", lw=1, linestyle="--")
plt.plot(fpr_train_set1, tpr_train_set1, label = "Train")
plt.plot(fpr_test_set1, tpr_test_set1, label = "Test")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
plt.show()
```



In [233]:

```
roc_auc_score(y_test,y_test_pred_proba_set1)
```

Out[233]:

```
0.6113295067472935
```

In [235]:

```
result_table['Hyper_Param_Depth'][0] = clf.best_estimator_.max_depth
result_table['Hyper_Param_Min_Sample_Split'][0] = clf.best_estimator_.min_samples_split
result_table['AUC'][0] = roc_auc_score(y_test,y_test_pred_proba_set1)
result_table
```

Out[235]:

	Vectorizer	Model	Hyper_Param_Depth	Hyper_Param_Min_Sample_Split	AUC
0	TFIDF	BRUTE	10.0	500.0	0.61133
1	TFIDFW2V	BRUTE	0.0	0.0	0.00000
2	TFIDF	Non_Zero_Features_Removed	0.0	0.0	0.00000
3	TFIDFW2V	Non_Zero_Features_Removed	0.0	0.0	0.00000

SET 2

DecisionTreeClassifier(max_depth=10, min_samples_split=500, random_state=42)

In [236]:

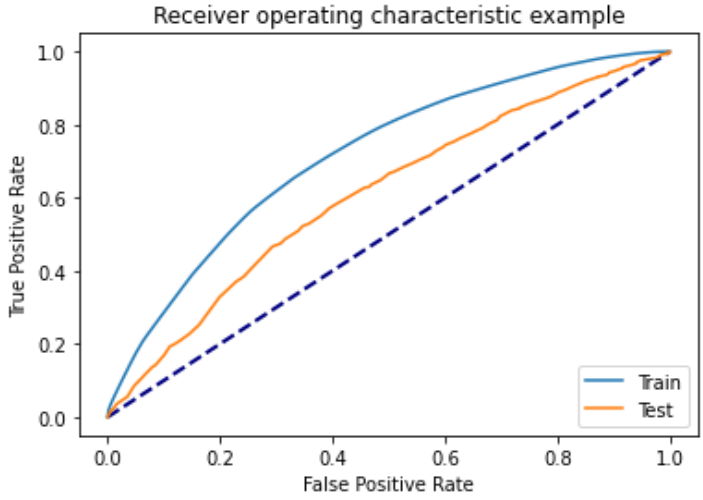
```
classif_set2 = DecisionTreeClassifier(max_depth = clf_set2.best_estimator_.max_depth , m
in_samples_split = clf_set2.best_estimator_.min_samples_split , random_state = clf_set2.
best_estimator_.random_state)
classif_set2.fit(X_tr_set2, y_train)
```

Out[236]:

```
DecisionTreeClassifier(max_depth=10, min_samples_split=500, random_state=42)
```

In [237]:

```
y_train_pred_set2 = classif_set2.predict(X_tr_set2)
y_test_pred_set2 = classif_set2.predict(X_te_set2)
y_train_pred_proba_set2 = classif_set2.predict_proba(X_tr_set2)[:,[1]]
y_test_pred_proba_set2 = classif_set2.predict_proba(X_te_set2)[:,[1]]
fpr_train_set2, tpr_train_set2, thresholds_train_set2 = roc_curve(y_train,y_train_pred_pr
oba_set2)
fpr_test_set2, tpr_test_set2, thresholds_test_set2 = roc_curve(y_test,y_test_pred_proba_s
et2)
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.plot(fpr_train_set2, tpr_train_set2, label = "Train")
plt.plot(fpr_test_set2, tpr_test_set2, label = "Test")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
plt.show()
```



In [238]:

In [238]:

```
result_table['Hyper_Param_Depth'][1] = clf_set2.best_estimator_.max_depth
result_table['Hyper_Param_Min_Sample_Split'][1] = clf_set2.best_estimator_.min_samples_sp
lit
result_table['AUC'][1] = roc_auc_score(y_test,y_test_pred_proba_set2)
result_table
```

Out[238]:

	Vectorizer	Model	Hyper_Param_Depth	Hyper_Param_Min_Sample_Split	AUC
0	TFIDF	BRUTE	10.0	500.0	0.611330
1	TFIDFW2V	BRUTE	10.0	500.0	0.609849
2	TFIDF	Non_Zero_Features_Removed	0.0	0.0	0.000000
3	TFIDFW2V	Non_Zero_Features_Removed	0.0	0.0	0.000000

step 11. Plot confusion matrix based on best threshold value

SET 1

In [240]:

```
# Pick the best threshold among the probability estimates, such that it has to yield maxi
mum value for TPR*(1-FPR)
# Plot the confusion matrices(each for train and test data) afer encoding the predicted c
lass labels, on the basis of the best threshod probability estimate.
```

In [239]:

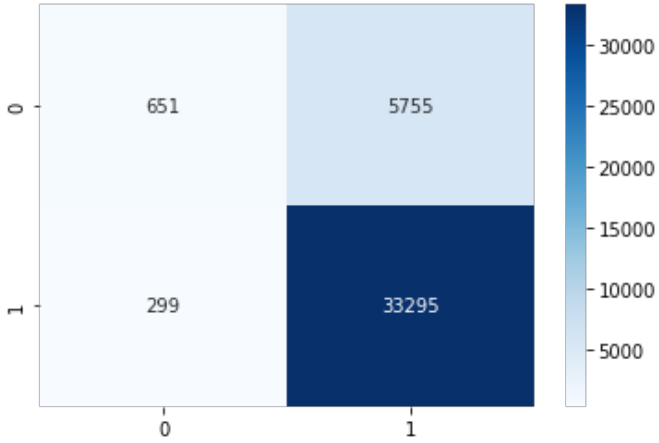
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round
(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [241]:

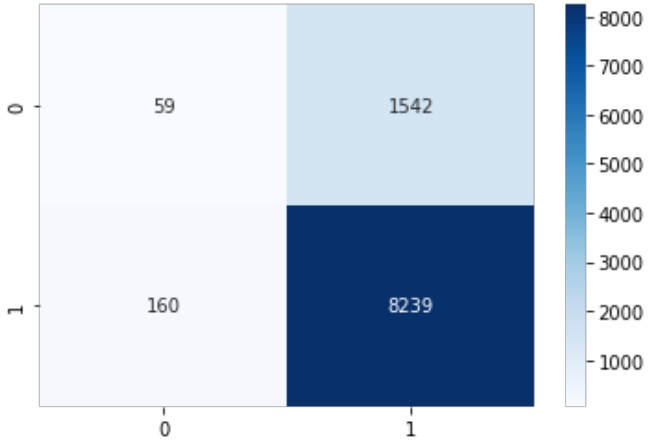
```
print("="*100)
best_t = find_best_threshold(thresholds_train_set1, fpr_train_set1, tpr_train_set1)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set1, best_t)))
sns.heatmap(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set1, best_t)),cma
p = 'Blues',annot = True,fmt = 'g')
plt.show()
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set1, best_t)))
sns.heatmap(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set1, best_t)),cmap
= 'Blues',annot = True,fmt = 'g')
plt.show()
```

```
====================================================================================
==========
```

```
the maximum value of tpr*(1-fpr) 0.3609585173199405 for threshold 0.857
Train confusion matrix
[[  651  5755]
 [  299 33295]]
```




```
Test confusion matrix
[[  59 1542]
 [ 160 8239]]
```

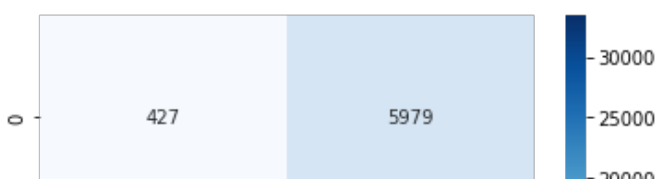


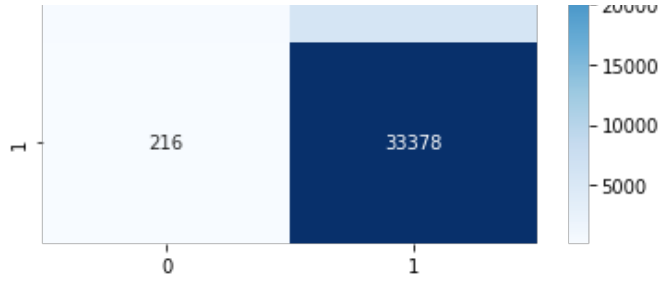
Set 2

In [242]:

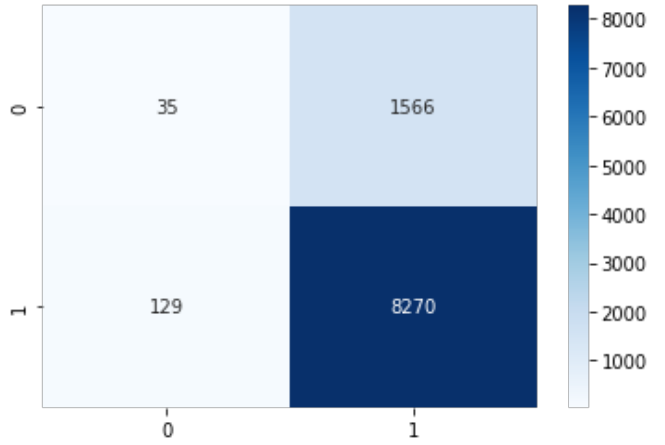
```
print("="*200)
best_t = find_best_threshold(thresholds_train_set2, fpr_train_set2, tpr_train_set2)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set2, best_t)))
sns.heatmap(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set2, best_t)),cma
p = 'Blues',annot = True,fmt = 'g')
plt.show()
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set2, best_t)))
sns.heatmap(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set2, best_t)),cmap
= 'Blues',annot = True,fmt = 'g')
plt.show()
```

```
========================================================================================
========================================================================================
======================
the maximum value of tpr*(1-fpr) 0.43737967532856536 for threshold 0.846
Train confusion matrix
[[  427  5979]
 [  216 33378]]
```






```
Test confusion matrix
[[  35 1566]
 [ 129 8270]]
```



Step 12. Find all the false positive data points and plot wordcloud of essay xt and pdf of teacher_number_of_previously_posted_projects.

set 1

In [243]:

```
y_test_set1 = y_test.copy()
y_test_set1['predicted'] = y_test_pred_set1
false_positive_points_essay = X_test.essay[y_test_set1.loc[(y_test_set1['project_is_appr
oved'] == 0) & (y_test_set1['predicted'] == 1)].index]
false_positive_points_essay
```

Out[243]:

```
0       my students title i setting they many challeng...
18      my students high poverty area diverse backgrou...
28      i work inner city school district our students...
39      my students come many diverse backgrounds rang...
47      my students awesome makers i service pre k 3 3...
                              ...
9985    the majority students i working come socioecon...
9995    in ms green class students challenged academic...
9996    my students attend school high poverty school ...
9998    you might not write well every day always edit...
9999    i 16 students composed 7 boys 9 girls there va...
Name: essay, Length: 1542, dtype: object
```

In [244]:

```
comment_words = ''
stopwords = set(STOPWORDS)
# iterate through the csv file
for val in false_positive_points_essay:
    # typecaste each val to string
    val = str(val)
    # split the value
```

```
    tokens = val.split()
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    comment_words += " ".join(tokens)+" "
wordcloud = WordCloud(width = 1000, height = 1000,background_color ='white',stopwords =
stopwords,min_font_size = 10).generate(comment_words)
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

set 2

In [245]:

```
y_test_set2 = y_test.copy()
y_test_set2['predicted'] = y_test_pred_set2
false_positive_points_essay_set2 = X_test.essay[y_test_set2.loc[(y_test_set2['project_is
_approved'] == 0) & (y_test_set2['predicted'] == 1)].index]
false_positive_points_essay_set2
```

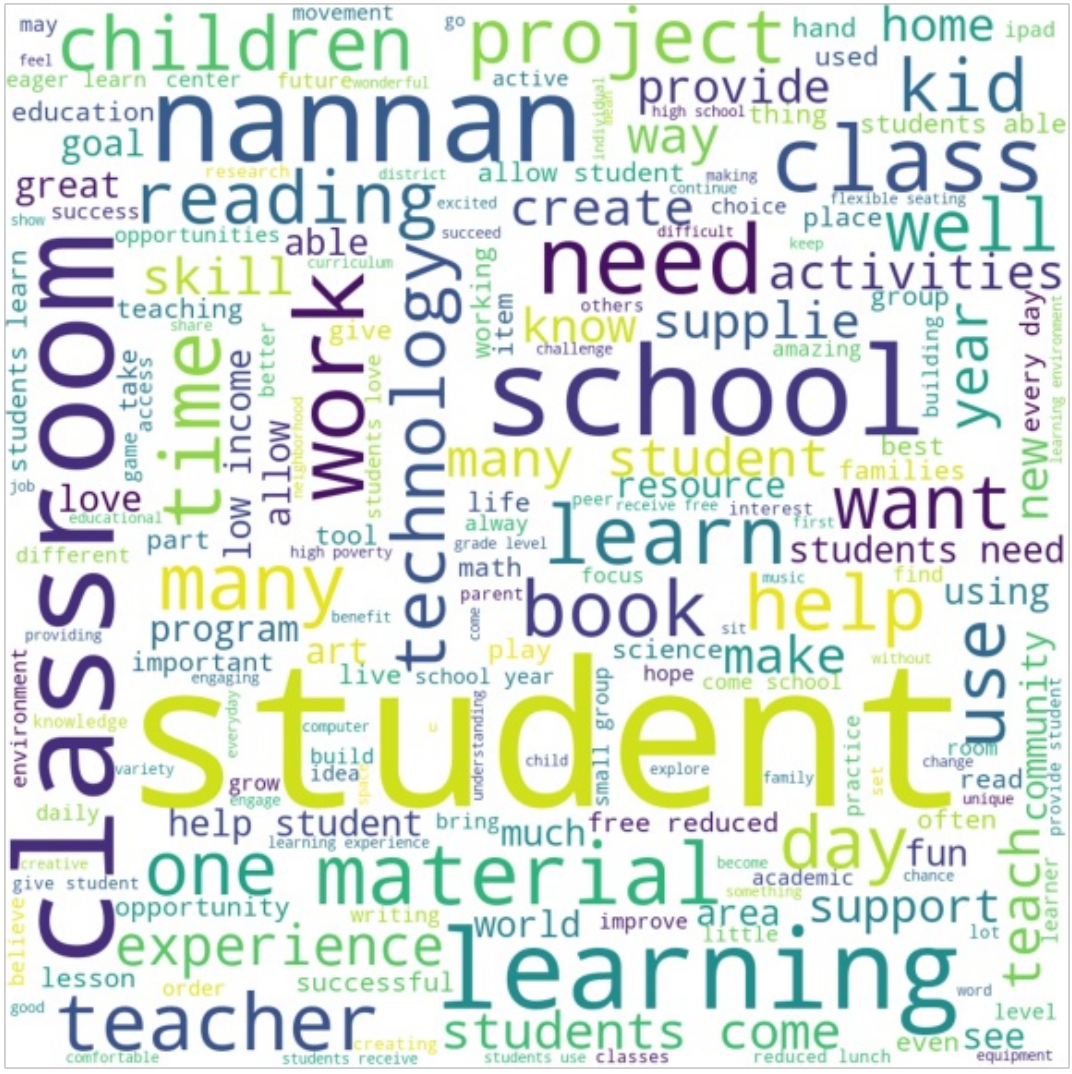
Out[245]:

```
0       my students title i setting they many challeng...
10      my students faced number challenges growing lo...
18      my students high poverty area diverse backgrou...
28      i work inner city school district our students...
39      my students come many diverse backgrounds rang...
                              ...
9985    the majority students i working come socioecon...
9995    in ms green class students challenged academic...
9996    my students attend school high poverty school...
```

```
9996    my students attend school high poverty school ...
9998    you might not write well every day always edit...
9999    i 16 students composed 7 boys 9 girls there va...
Name: essay, Length: 1566, dtype: object
```

In [246]:

```
comment_words = ''
stopwords = set(STOPWORDS)
# iterate through the csv file
for val in false_positive_points_essay_set2:
    # typecaste each val to string
    val = str(val)
    # split the value
    tokens = val.split()
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    comment_words += " ".join(tokens)+" "
wordcloud = WordCloud(width = 1000, height = 1000,background_color ='white',stopwords =
stopwords,min_font_size = 10).generate(comment_words)
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Plot the box plot with the `price` of these `false positive data points`

set 1

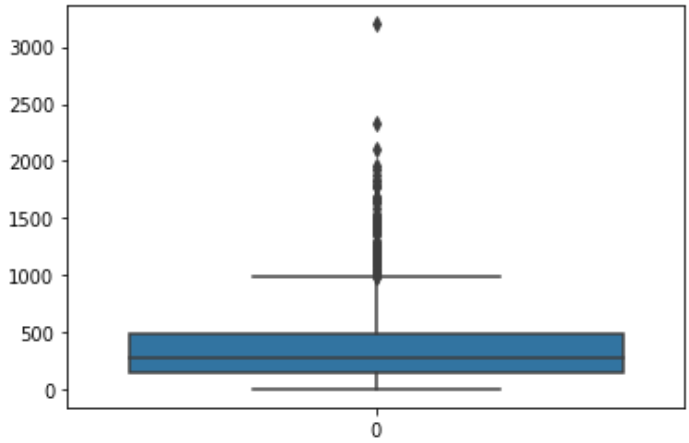
In [247]:

In [247]:

```
false_positive_points_price = X_test.price[y_test_set1.loc[(y_test_set1['project_is_approved'] == 0) & (y_test_set1['predicted'] == 1)].index]
sns.boxplot(data=false_positive_points_price)
```

Out[247]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4597c75290>
```



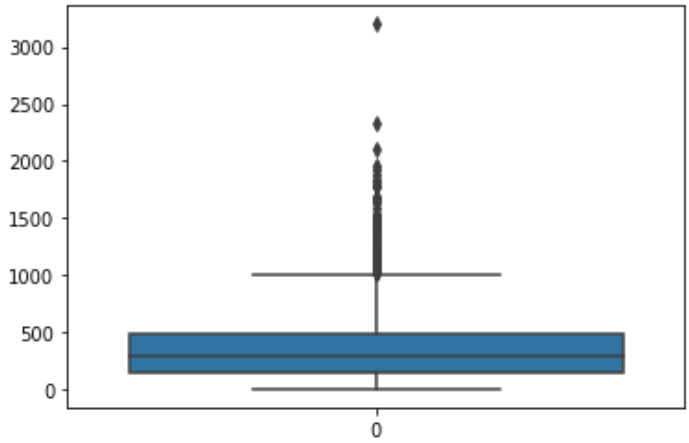
set2

In [248]:

```
false_positive_points_price_set2 = X_test.price[y_test_set2.loc[(y_test_set2['project_is_approved'] == 0) & (y_test_set2['predicted'] == 1)].index]
sns.boxplot(data=false_positive_points_price_set2)
```

Out[248]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4598eebb50>
```



Plot the pdf with the `teacher number_of_previously_posted_projects` of these `false positive data points`

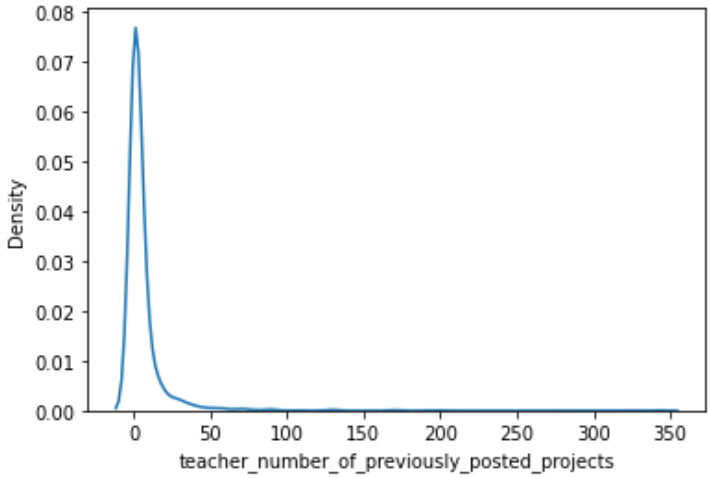
set 1

In [249]:

```
false_positive_points_teacher_number_of_previously_posted_projects = X_test.teacher_number_of_previously_posted_projects[y_test_set1.loc[(y_test_set1['project_is_approved'] == 0) & (y_test_set1['predicted'] == 1)].index]
sns.kdeplot(data=false_positive_points_teacher_number_of_previously_posted_projects)
```

Out[249]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4594a6f6d0>
```



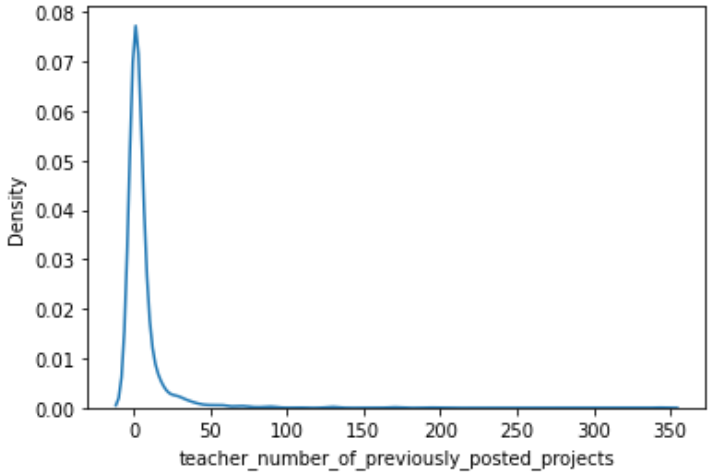
set 2

In [251]:

```
false_positive_points_teacher_number_of_previously_posted_projects_set2 = X_test.teacher_
number_of_previously_posted_projects[y_test_set2.loc[(y_test_set2['project_is_approved']
== 0) & (y_test_set2['predicted'] == 1)].index]
sns.kdeplot(data=false_positive_points_teacher_number_of_previously_posted_projects_set2)
```

Out[251]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f45993dcf10>
```



Step 13. Write your observations about the wordcloud and pdf.

In [252]:

```
#wordcloud.words_
```

Word Cloud

- **words like student,classroom,school,learning,work,need,class,material are dominating and these words are influencing the essays to get funding.**
- **Our model learnt that above words if present give more weightage towards approval of the project.**

box plot

- **We can see that most the False positive points have price in the range around 50 to 500 that means our model learnt that when ever price of the project is low it is approving the project.**

cdf plot

- **here model learnt that when ever the number of teacher_number_of_previously_posted_projects is low our model is approving the project for most of the false positive points.**

Note : Observations are same for tf-idf and tf-idf w2v embeddings

Task - 2

In [253]:

```
# 1. write your code in following steps for task 2
# 2. select all non zero features
# 3. Update your dataset i.e. X_train,X_test and X_cv so that it contains all rows and on
ly non zero features
# 4. perform hyperparameter tuning and plot either heatmap or 3d plot.
# 5. Fit the best model. Plot ROC AUC curve and confusion matrix similar to model 1.
```

2. select all non zero features

set 1

In [254]:

```
X_tr_set1 =  X_tr_set1.toarray()
X_tr_set1_non_zero = X_tr_set1[:,np.where(classif_set1.feature_importances_ != 0)[0]]
X_te_set1_non_zero = X_te_set1[:,np.where(classif_set1.feature_importances_ != 0)[0]]
```

In [255]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_nonzero_grid.pkl
'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_nonzero_grid.pkl', 'rb
') as file:
    clf = pickle.load(file)
else:
  classif_nonzero_set1 = DecisionTreeClassifier(random_state = 42)
  parameters = {'max_depth': [1, 3, 10, 30],'min_samples_split': [5, 10, 100, 500]}
  clf = GridSearchCV(classif_nonzero_set1, parameters, cv=5, scoring='roc_auc',return_tr
ain_score=True)
  clf.fit(X_tr_set1_non_zero, y_train)
```

In [256]:

```
if not os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set1_nonzero_grid
.pkl'):
  with open('set1_nonzero_grid.pkl', 'wb') as file:
    pickle.dump(clf, file)
  !cp set1_nonzero_grid.pkl /content/drive/MyDrive/9_Donors_choose_DT/models
```

In [257]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results.head(5)
```

Out[257]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	par
0	0.102553	0.010457	0.007229	0.000598	1	5	{'max_depth 'min_samples_s
							{'max_depth

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	par
1	0.101651	0.003888	0.007288	0.000571	1	10	'min_samples_s
2	0.098018	0.004142	0.006844	0.000264	1	100	{'max_depth 'min_samples_s
3	0.097803	0.002028	0.006689	0.000048	1	500	{'max_depth 'min_samples_s
4	0.236616	0.015394	0.007210	0.000193	3	5	{'max_depth 'min_samples_s

5 rows × 22 columns

In [258]:

```
result_required_non_zero_set1 = results[['param_max_depth', 'param_min_samples_split','mean_test_score','mean_train_score']]
result_required_non_zero_set1.head(5)
```

Out[258]:

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score
0	1	5	0.548110	0.550340
1	1	10	0.548110	0.550340
2	1	100	0.548110	0.550340
3	1	500	0.548110	0.550340
4	3	5	0.584869	0.594896

3D scatter plot

In [259]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=result_required_non_zero_set1['param_max_depth'],y=result_required_non_zero_set1['param_min_samples_split'],z=result_required_non_zero_set1['mean_train_score'], name = 'train')
trace2 = go.Scatter3d(x=result_required_non_zero_set1['param_max_depth'],y=result_required_non_zero_set1['param_min_samples_split'],z=result_required_non_zero_set1['mean_test_score'], name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(width=1000,
    height=1000,scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
fig.show(renderer = 'colab')
plt.show()
```

Heat Map

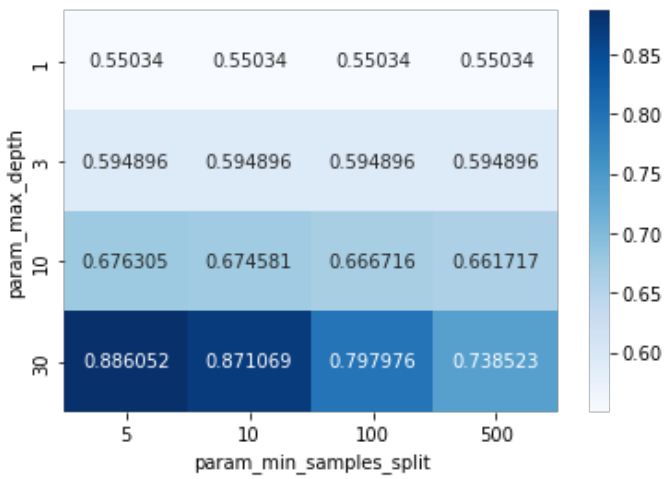
Heat Map For Train

In [260]:

```
sns.heatmap(result_required_non_zero_set1.pivot('param_max_depth', 'param_min_samples_split', 'mean_train_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[260]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459a57f890>
```



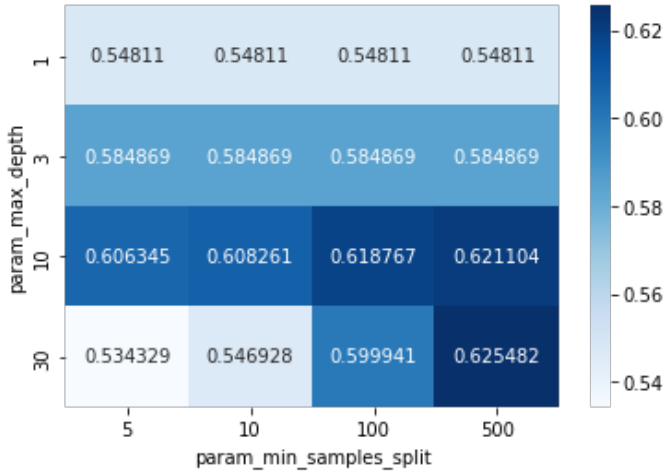
Heat Map For Test

In [261]:

```
sns.heatmap(result_required_non_zero_set1.pivot('param_max_depth', 'param_min_samples_split', 'mean_test_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[261]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4597f79d50>
```



In [262]:

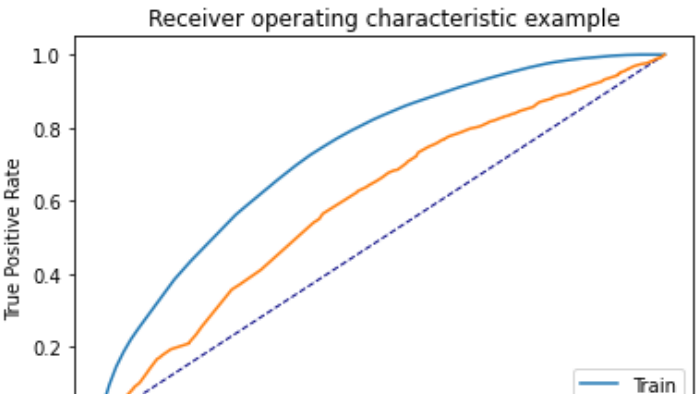
```
classif_nonzero_set1 = DecisionTreeClassifier(max_depth = clf.best_estimator_.max_depth , min_samples_split = clf.best_estimator_.min_samples_split , random_state = clf.best_estimator_.random_state)
classif_nonzero_set1.fit(X_tr_set1_non_zero, y_train)
```

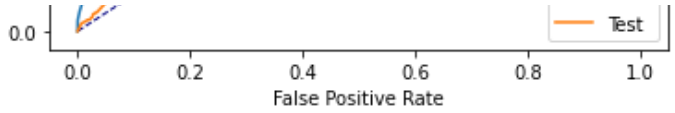
Out[262]:

```
DecisionTreeClassifier(max_depth=30, min_samples_split=500, random_state=42)
```

In [263]:

```
y_train_pred_set1 = classif_nonzero_set1.predict(X_tr_set1_non_zero)
y_test_pred_set1 = classif_nonzero_set1.predict(X_te_set1_non_zero)
y_train_pred_proba_set1 = classif_nonzero_set1.predict_proba(X_tr_set1_non_zero)[:,[1]]
y_test_pred_proba_set1 = classif_nonzero_set1.predict_proba(X_te_set1_non_zero)[:,[1]]
fpr_train_set1, tpr_train_set1, thresholds_train_set1 = roc_curve(y_train,y_train_pred_proba_set1)
fpr_test_set1, tpr_test_set1, thresholds_test_set1 = roc_curve(y_test,y_test_pred_proba_set1)
plt.plot([0, 1], [0, 1], color="navy", lw=1, linestyle="--")
plt.plot(fpr_train_set1, tpr_train_set1, label = "Train")
plt.plot(fpr_test_set1, tpr_test_set1, label = "Test")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
plt.show()
```



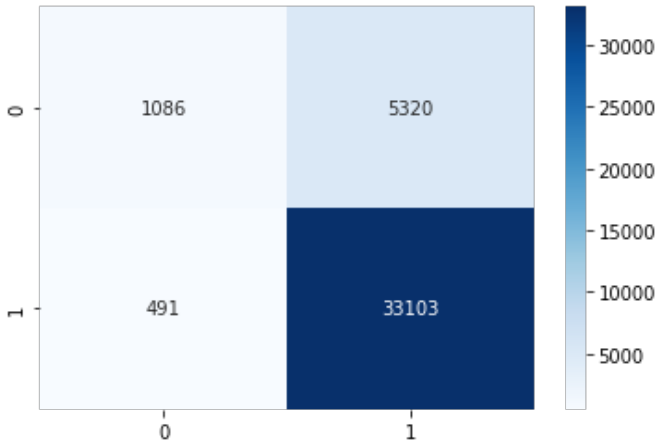


In [264]:

```
sns.heatmap(confusion_matrix(y_train,y_train_pred_set1),annot =True , cmap = 'Blues',fmt = 'g')
```

Out[264]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459a1c7190>
```

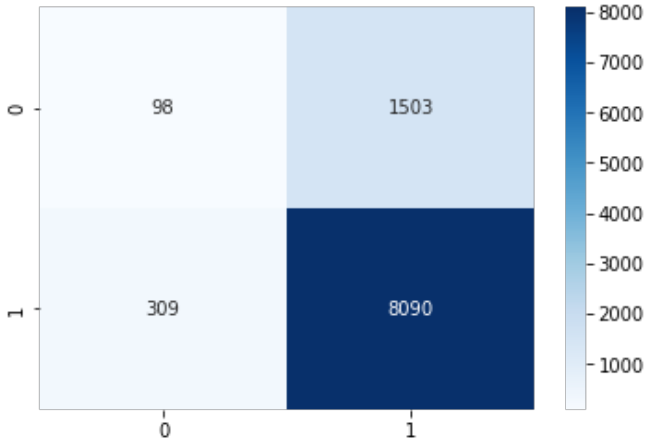


In [265]:

```
sns.heatmap(confusion_matrix(y_test,y_test_pred_set1),annot =True , cmap = 'Blues',fmt = 'g')
```

Out[265]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f45996bdc10>
```



In [266]:

```
result_table['Hyper_Param_Depth'][2] = clf.best_estimator_.max_depth
result_table['Hyper_Param_Min_Sample_Split'][2] = clf.best_estimator_.min_samples_split
result_table['AUC'][2] = roc_auc_score(y_test,y_test_pred_proba_set1)
result_table
```

Out[266]:

	Vectorizer	Model	Hyper_Param_Depth	Hyper_Param_Min_Sample_Split	AUC
0	TFIDF	BRUTE	10.0	500.0	0.611330
1	TFIDFW2V	BRUTE	10.0	500.0	0.609849
2	TFIDF	Non_Zero_Features_Removed	30.0	500.0	0.603897
3	TFIDFW2V	Non_Zero_Features_Removed	0.0	0.0	0.000000

set 2

In [267]:

```
X_tr_set2 =  X_tr_set2.toarray()
X_tr_set2_non_zero = X_tr_set2[:,np.where(classif_set2.feature_importances_ != 0)[0]]
X_te_set2_non_zero = X_te_set2[:,np.where(classif_set2.feature_importances_ != 0)[0]]
```

In [268]:

```
if os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_nonzero_grid.pkl
'):
  with open('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_nonzero_grid.pkl', 'rb
') as file:
    clf = pickle.load(file)
else:
  classif_nonzero_set2 = DecisionTreeClassifier(random_state = 42)
  parameters = {'max_depth': [1, 3, 10, 30],'min_samples_split': [5, 10, 100, 500]}
  clf = GridSearchCV(classif_nonzero_set2, parameters, cv=5, scoring='roc_auc',return_tr
ain_score=True)
  clf.fit(X_tr_set2_non_zero, y_train)
```

In [269]:

```
if not os.path.exists('/content/drive/MyDrive/9_Donors_choose_DT/models/set2_nonzero_grid
.pkl'):
  with open('set2_nonzero_grid.pkl', 'wb') as file:
    pickle.dump(clf, file)
  !cp set2_nonzero_grid.pkl /content/drive/MyDrive/9_Donors_choose_DT/models
```

In [270]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results.head(5)
```

Out[270]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	par
0	0.670414	0.008791	0.007589	0.000173	1	5	{'max_depth 'min_samples_s
1	0.664438	0.005419	0.007599	0.000210	1	10	{'max_depth 'min_samples_s
2	0.690348	0.048002	0.007862	0.000234	1	100	{'max_depth 'min_samples_s
3	0.668305	0.008820	0.007662	0.000181	1	500	{'max_depth 'min_samples_s
4	1.845902	0.017879	0.008914	0.001608	3	5	{'max_depth 'min_samples_s

5 rows × 22 columns

In [271]:

```
result_required_non_zero_set2 = results[['param_max_depth', 'param_min_samples_split','me
an_test_score','mean_train_score']]
result_required_non_zero_set2.head(5)
```

Out[271]:

	param_max_depth	param_min_samples_split	mean_test_score	mean_train_score
0	1	5	0.546486	0.556617
1	1	10	0.546486	0.556617
2	1	100	0.546486	0.556617
3	1	500	0.546486	0.556617
4	3	5	0.591417	0.613803

3D scatter plot

In [272]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=result_required_non_zero_set2['param_max_depth'],y=result_requir
ed_non_zero_set2['param_min_samples_split'],z=result_required_non_zero_set2['mean_train_s
core'], name = 'train')
trace2 = go.Scatter3d(x=result_required_non_zero_set2['param_max_depth'],y=result_requir
ed_non_zero_set2['param_min_samples_split'],z=result_required_non_zero_set2['mean_test_sc
ore'], name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(width=1000,
    height=1000,scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
fig.show(renderer = 'colab')
plt.show()
```

Heat Map

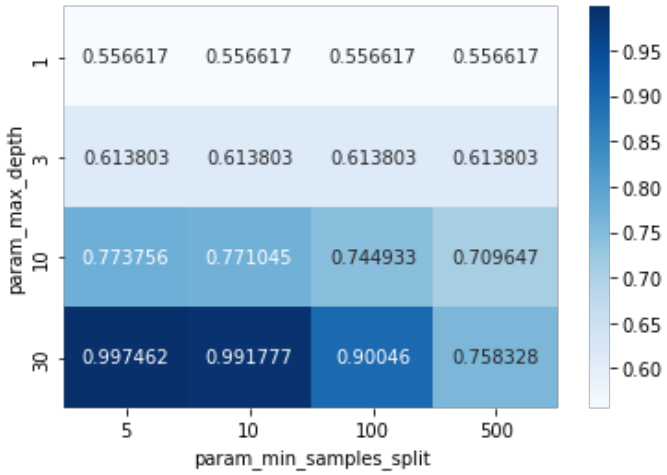
Heat Map For Train

In [273]:

```
sns.heatmap(result_required_non_zero_set2.pivot('param_max_depth', 'param_min_samples_split', 'mean_train_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[273]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459a1e42d0>
```



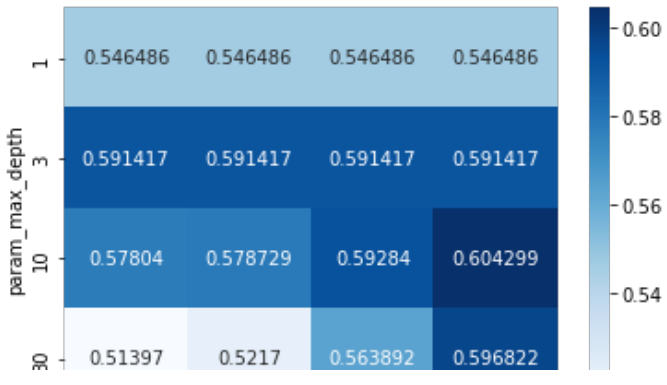
Heat Map For Test

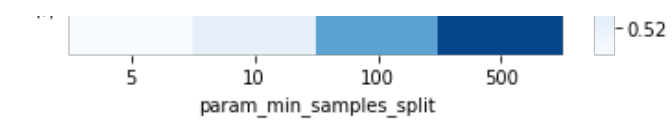
In [274]:

```
sns.heatmap(result_required_non_zero_set2.pivot('param_max_depth', 'param_min_samples_split', 'mean_test_score') ,annot=True,cmap = 'Blues',fmt='g')
```

Out[274]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f45994fea90>
```





In [275]:

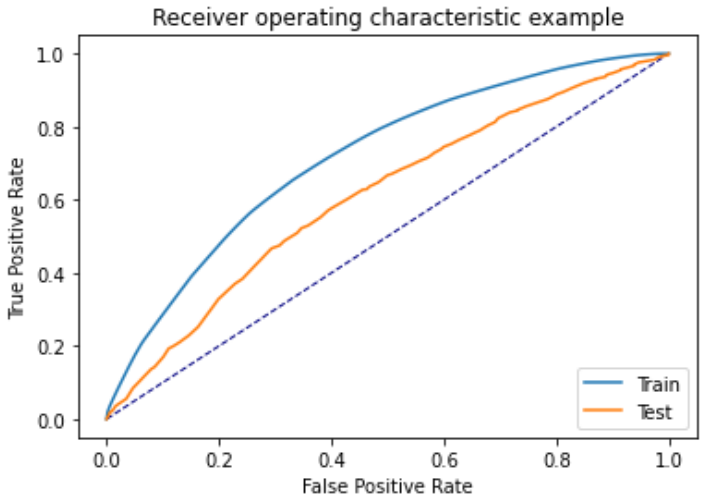
```
classif_nonzero_set2 = DecisionTreeClassifier(max_depth = clf.best_estimator_.max_depth ,
min_samples_split = clf.best_estimator_.min_samples_split , random_state = clf.best_esti
mator_.random_state)
classif_nonzero_set2.fit(X_tr_set2_non_zero, y_train)
```

Out[275]:

```
DecisionTreeClassifier(max_depth=10, min_samples_split=500, random_state=42)
```

In [276]:

```
y_train_pred_set2 = classif_nonzero_set2.predict(X_tr_set2_non_zero)
y_test_pred_set2 = classif_nonzero_set2.predict(X_te_set2_non_zero)
y_train_pred_proba_set2 = classif_nonzero_set2.predict_proba(X_tr_set2_non_zero)[:,[1]]
y_test_pred_proba_set2 = classif_nonzero_set2.predict_proba(X_te_set2_non_zero)[:,[1]]
fpr_train_set2, tpr_train_set2, thresholds_train_set2 = roc_curve(y_train,y_train_pred_pr
oba_set2)
fpr_test_set2, tpr_test_set2, thresholds_test_set2 = roc_curve(y_test,y_test_pred_proba_s
et2)
plt.plot([0, 1], [0, 1], color="navy", lw=1, linestyle="--")
plt.plot(fpr_train_set2, tpr_train_set2, label = "Train")
plt.plot(fpr_test_set2, tpr_test_set2, label = "Test")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
plt.show()
```

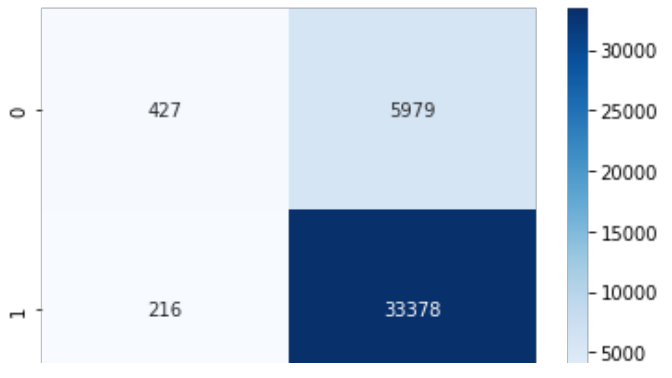


In [277]:

```
sns.heatmap(confusion_matrix(y_train,y_train_pred_set2),annot =True , cmap = 'Blues',fmt
= 'g')
```

Out[277]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4597f54ed0>
```

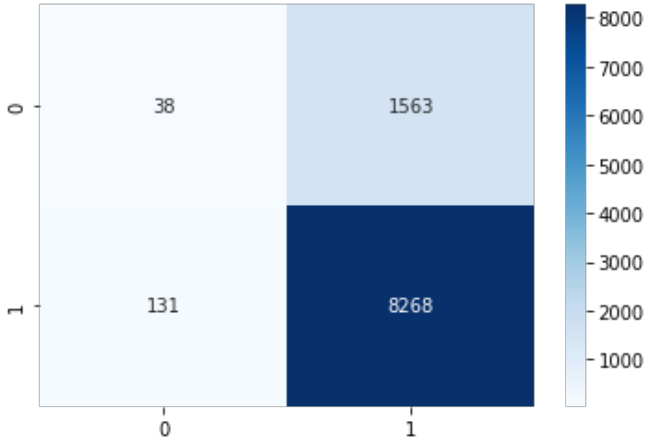


In [278]:

```
sns.heatmap(confusion_matrix(y_test,y_test_pred_set2),annot =True , cmap = 'Blues',fmt =
'g')
```

Out[278]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459ad05790>
```



In [279]:

```
result_table['Hyper_Param_Depth'][3] = clf.best_estimator_.max_depth
result_table['Hyper_Param_Min_Sample_Split'][3] = clf.best_estimator_.min_samples_split
result_table['AUC'][3] = roc_auc_score(y_test,y_test_pred_proba_set2)
result_table
```

Out[279]:

	Vectorizer	Model	Hyper_Param_Depth	Hyper_Param_Min_Sample_Split	AUC
0	TFIDF	BRUTE	10.0	500.0	0.611330
1	TFIDFW2V	BRUTE	10.0	500.0	0.609849
2	TFIDF	Non_Zero_Features_Removed	30.0	500.0	0.603897
3	TFIDFW2V	Non_Zero_Features_Removed	10.0	500.0	0.610476

In [81]:

```
# Tabulate your results
```

In [293]:

```
from tabulate import tabulate
print(tabulate(result_table, headers=result_table.columns,showindex = 'never',tablefmt='
fancy_grid'))
```

```
╒═══════════════╤═══════════════════════════╤═══════════════════════╤═══════════════════════════
╤═══════╕
│ Vectorizer    │ Model                     │   Hyper_Param_Depth   │   Hyper_Param_Min_Sam
ple_Split │   AUC │
╞═══════════════╪═══════════════════════════╪═══════════════════════╪═══════════════════════════
╪═══════╡
│ TFIDF         │ BRUTE                     │                    10 │
500 │ 0.61133 │
├───────────────┼───────────────────────────┼───────────────────────┼───────────────────────────
┼───────┤
│ TFIDFW2V      │ BRUTE                     │                    10 │
500 │ 0.609849 │
├───────────────┼───────────────────────────┼───────────────────────┼───────────────────────────
┼───────┤
│ TFIDF         │ Non_Zero_Features_Removed │                    30 │
```

```
500 | 0.603897 |
├──────────────┼──────────────────────────┼──────────────────────┼──────────────┤
| TFIDFW2V     | Non_Zero_Features_Removed |                   10 |
500 | 0.610476 |
└──────────────┴──────────────────────────┴──────────────────────┴──────────────┘
```

In [ ]: