Assignment : DT

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 [72]:
```
#please use below code to load glove vectors
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

or else , you can use below code

In [73]:
```
'''
# 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-var

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


'''
```

Out[73]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(gloveFile)
:\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for lin
e in tqdm(f):\n        splitLine = line.split()\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 preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n
```

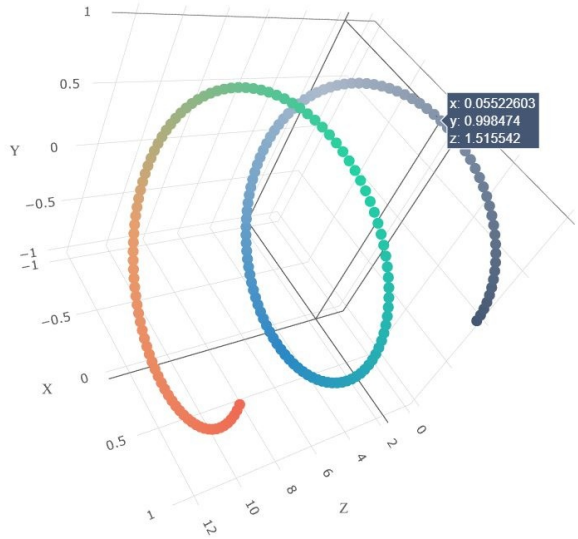
```
words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the un
ique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number
of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(inter_wo
rds)/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(words_courpus))\n\n\n# strongin
g variables into pickle files python: http://www.jessicayung.com/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, 5, 10, 50], 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)

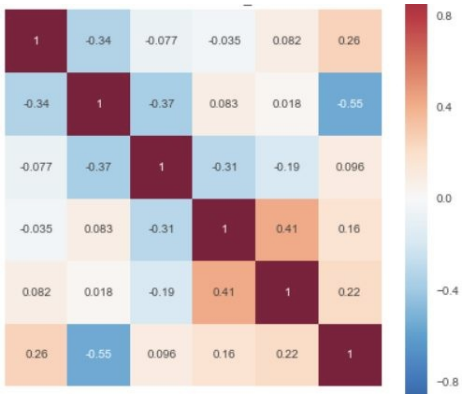
     - Representation of results
       - You need to plot the performance of model both on train data and cross validation data for each hyper parameter,



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*

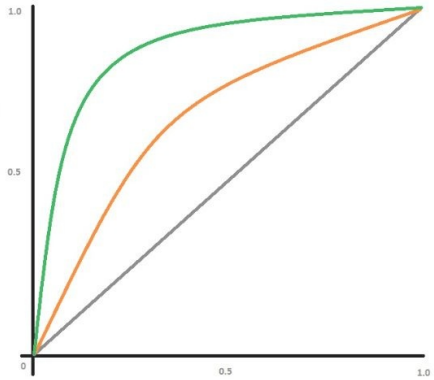
or

       - 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.

- 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`

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 [75]:
```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to C:\Users\Subhashini
[nltk_data]     Rajesh\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[75]: True

In [76]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with the biggest
for learning my students learn in many different ways using all of our senses and multiple intelligences i use a
of techniques to help all my students succeed students in my class come from a variety of different backgrounds w
for wonderful sharing of experiences and cultures including native americans our school is a caring community of
learners which can be seen through collaborative student project based learning in and out of the classroom kinde
in my class love to work with hands on materials and have many different opportunities to practice a skill before
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curr
montana is the perfect place to learn about agriculture and nutrition my students love to role play in our preter
in the early childhood classroom i have had several kids ask me can we try cooking with real food i will take the
and create common core cooking lessons where we learn important math and writing concepts while cooking delicious
food for snack time my students will have a grounded appreciation for the work that went into making the food and
of where the ingredients came from as well as how it is healthy for their bodies this project would expand our le
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make our
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks to be pri
```

```
shared with families students will gain math and literature skills as well as a life long enjoyment for healthy c
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

TASK - 1

1. Decision Tree

In [300…

```
import warnings
warnings.filterwarnings("ignore")

import pickle
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
import scipy
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import seaborn as sns
from sklearn.metrics import roc_curve, auc
from wordcloud import WordCloud, STOPWORDS
from sklearn.linear_model import LogisticRegression
```

In [ ]:

```
```

1.1 Loading Data

In [78]:

```
import pandas
data = pandas.read_csv('preprocessed_data.csv')
data.head()
```

Out[78]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	clean_categories	cl
0	ca	mrs	grades_prek_2	53	1	math_science	
1	ut	ms	grades_3_5	4	1	specialneeds	
2	ca	mrs	grades_prek_2	10	1	literacy_language	
3	ga	mrs	grades_prek_2	2	1	appliedlearning	
4	wa	mrs	grades_3_5	2	1	literacy_language	

In [79]:
```
# Sentiment Analysis on 'essay'
sid    = SentimentIntensityAnalyzer()

negative_sentiments = []
positive_sentiments = []
neutral_sentiments = []
compound_sentiments = []

for i in tqdm(data['essay']):
  sid_sentiments = sid.polarity_scores(i)
  negative_sentiments.append(sid_sentiments['neg'])
  positive_sentiments.append(sid_sentiments['pos'])
  neutral_sentiments.append(sid_sentiments['neu'])
  compound_sentiments.append(sid_sentiments['compound'])

# Now append these sentiments columns/features to original preprocessed dataframe
data['negative_sent'] = negative_sentiments
data['positive_sent'] = positive_sentiments
data['neutral_sent'] = neutral_sentiments
data['compound_sent'] = compound_sentiments
```

```
100%|██████████| 109248/109248 [04:44<00:00, 383.65it/s]
```

1.1 Splitting a data

In [80]:
```
# Sepearting input data and labels to have a proper data-matrix
Y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
X_train['negative_sent'].shape
```

```
(76473, 12) (76473,)
(32775, 12) (32775,)
```

Out[80]:
```
(76473,)
```

1.2 One hot encoding of categorical features

In [81]:
```
#seperating categorical features alone
#Y_cat = data['project_is_approved'].values
#X_cat = data.drop(['project_is_approved', 'teacher_number_of_previously_posted_projects', 'price' ], axis=1)

#X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(X_cat, Y_cat, test_size=0.3)

#print(X_train_cat.shape, y_train_cat.shape)
#print(X_test_cat.shape, y_test_cat.shape)
```

In [82]:
```
encoder = OneHotEncoder(handle_unknown = 'ignore', sparse = False)
```

In [83]:
```
# Categorical features are school_state, teacher_prefix, project_grade_category, clean_categories,clean_subcatego
```

In [84]:
```
cat_data = ['school_state', 'teacher_prefix', 'project_grade_category', 'clean_categories','clean_subcategories']
for i in cat_data:
    encoder.fit(X_train[i].values.reshape(-1,1))
#     cat_x_trn = encoder.transform(X_train[i].values.reshape(-1,1))
#     cat_x_tst = encoder.transform(X_test[i].values.reshape(-1,1))
#     np.hstack(cat_x_trn, cat_x_tst)

# OneHotEncoding of feature school state
state_xtrn = encoder.transform(X_train['school_state'].values.reshape(-1,1))
state_xtst = encoder.transform(X_test['school_state'].values.reshape(-1,1))

# OneHotEncoding of feature teacher prefix
teacher_xtrn = encoder.transform(X_train['teacher_prefix'].values.reshape(-1,1))
teacher_xtst = encoder.transform(X_test['teacher_prefix'].values.reshape(-1,1))

# OneHotEncoding of feature school state project grade category
prj_grade_xtrn = encoder.transform(X_train['project_grade_category'].values.reshape(-1,1))
prj_grade_xtst = encoder.transform(X_test['project_grade_category'].values.reshape(-1,1))

# OneHotEncoding of feature school state clean categries
cln_cat_xtrn = encoder.transform(X_train['clean_categories'].values.reshape(-1,1))
cln_cat_xtst = encoder.transform(X_test['clean_categories'].values.reshape(-1,1))

# OneHotEncoding of feature school state clean subcategories
cln_subcat_xtrn = encoder.transform(X_train['clean_subcategories'].values.reshape(-1,1))
cln_subcat_xtst = encoder.transform(X_test['clean_subcategories'].values.reshape(-1,1))
```

1.3 Scaling on numerical feature

In [85]:
```
# the numerical features are  teacher_number_of_previously_posted_projects      price
```

In [86]:
```
normalizer = Normalizer()
```

In [87]:
```
num_data = ['price', 'teacher_number_of_previously_posted_projects', 'negative_sent', 'positive_sent', 'neutral_s
for j in num_data:
    normalizer.fit(X_train[j].values.reshape(-1,1))

# scaling the numeric feature price
X_train_normalized_price = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_test_normalized_price = normalizer.transform(X_test['price'].values.reshape(-1,1))

#  scaling the numeric feature prev posted projects
X_train_normalized_proj = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.res
X_test_normalized_proj = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.resha

# scaling the numeric feature negative sent
X_train_normalized_neg = normalizer.transform(X_train['negative_sent'].values.reshape(-1,1))
X_test_normalized_neg = normalizer.transform(X_test['negative_sent'].values.reshape(-1,1))

# scaling the numeric feature positive sent
X_train_normalized_pos = normalizer.transform(X_train['positive_sent'].values.reshape(-1,1))
X_test_normalized_pos = normalizer.transform(X_test['positive_sent'].values.reshape(-1,1))

# scaling the numeric feature neutral sent
X_train_normalized_neu = normalizer.transform(X_train['neutral_sent'].values.reshape(-1,1))
X_test_normalized_neu = normalizer.transform(X_test['neutral_sent'].values.reshape(-1,1))

# scaling the numeric feature compound sent
X_train_normalized_com = normalizer.transform(X_train['compound_sent'].values.reshape(-1,1))
X_test_normalized_com= normalizer.transform(X_test['compound_sent'].values.reshape(-1,1))
```

In [88]:
```
print('After Normalizing on price column checking the shapes ')
print(X_train_normalized_price.shape, y_train.shape)
print(X_test_normalized_price.shape, y_test.shape)
```

```
After Normalizing on price column checking the shapes
(76473, 1) (76473,)
(32775, 1) (32775,)
```

In [89]:
```
print('After Normalizing on num of project column checking the shapes ')
print(X_train_normalized_proj.shape, y_train.shape)
print(X_test_normalized_proj.shape, y_test.shape)
```

```
After Normalizing on num of project column checking the shapes
(76473, 1) (76473,)
(32775, 1) (32775,)
```

In [90]:
```
print('After Normalizing on neg sent column checking the shapes ')
print(X_train_normalized_neg.shape, y_train.shape)
print(X_test_normalized_neg.shape, y_test.shape)
```

```
After Normalizing on neg sent column checking the shapes
(76473, 1) (76473,)
(32775, 1) (32775,)
```

1.4 Essay to vector (TFIDF)

In [91]:
```
#applying TFIDF for essay column
vec_essay_tfidf = TfidfVectorizer()
#call fit func only for X_train
vec_essay_tfidf.fit(X_train['essay'].values)

#convert essay to vectors
xtrn_vec_essay_tfidf = vec_essay_tfidf.transform(X_train['essay'].values)
xtst_vec_essay_tfidf = vec_essay_tfidf.transform(X_test['essay'].values)

print('After TFIDF on Essay column checking the shapes ')
print(xtrn_vec_essay_tfidf.shape, y_train.shape)
print(xtst_vec_essay_tfidf.shape, y_test.shape)
```

```
After TFIDF on Essay column checking the shapes
(76473, 49127) (76473,)
(32775, 49127) (32775,)
```

1.5 Essay to vector (TFIDF W2V)

In [92]:
```
# Reference_Vectorization from AAIC
# average Word2Vec
# compute average word2vec for each review.
def tfidfw2v(data):
```

```
    avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(data): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        cnt_words =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:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors.append(vector)
    return np.array(avg_w2v_vectors)
```

In [93]:
```
X_train_tfidfw2v = tfidfw2v(X_train['essay'].values)
X_test_tfidfw2v = tfidfw2v(X_test['essay'].values)
```

```
100%|██████████| 76473/76473 [00:23<00:00, 3249.41it/s]
100%|██████████| 32775/32775 [00:11<00:00, 2847.46it/s]
```

In [108…
```
print(X_train_tfidfw2v.shape)
print(X_test_tfidfw2v.shape)
```

```
(76473, 300)
(32775, 300)
```

Set 1: categorical, numerical features + preprocessed_essay (TFIDF) + Sentiment scores(preprocessed_essay)

In [101…
```
#SET1 X_train
X_train_set1 = scipy.sparse.hstack((state_xtrn, teacher_xtrn, prj_grade_xtrn, cln_cat_xtrn, cln_subcat_xtrn, X_tr
```

In [96]:
```
#SET1 X_test
X_test_set1 = scipy.sparse.hstack((state_xtst, teacher_xtst, prj_grade_xtst, cln_cat_xtst, cln_subcat_xtst, X_tes
```

Set 2: categorical, numerical features + preprocessed_essay (TFIDFW2V) + Sentiment scores(preprocessed_essay)

In [99]:
```
#SET2 X_train
X_trn_set2 = np.hstack((state_xtrn, teacher_xtrn, prj_grade_xtrn, cln_cat_xtrn, cln_subcat_xtrn, X_train_normaliz
```

In [109…
```
#SET2 X_test
X_tst_set2 = np.hstack((state_xtst, teacher_xtst, prj_grade_xtst, cln_cat_xtst, cln_subcat_xtst, X_test_normalize
```

In [ ]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# 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 code
# 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
```

Hyperparameter Tuning Set1 features with GridSearchCV

In [115…
```
depth = [1, 5, 10, 50]
sample_split =  [5, 10, 100, 500]
params = {'max_depth' : depth, 'min_samples_split' : sample_split}
DT = DecisionTreeClassifier(class_weight = 'balanced')
model_s1 = GridSearchCV(DT, params, cv = 3, scoring = 'roc_auc', return_train_score = True)
model_s1.fit(X_train_set1, y_train)
print('Best Paramters:', model_s1.best_params_)
```

```
Best Paramters: {'max_depth': 10, 'min_samples_split': 500}
```

In [165…
```
results = pd.DataFrame.from_dict(model_s1.cv_results_)
# results = results.sort_values(['mean_train_score', 'mean_test_score'])
```

In [166…
```
results.head()
```

Out[166…

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_split	params	split0_test_s
0	2.598533	0.087006	0.042156	0.004417	1	5	{'max_depth': 1, 'min_samples_split': 5}	0.55
1	2.688673	0.058456	0.033692	0.002609	1	10	{'max_depth': 1, 'min_samples_split': 10}	0.55
							{'max_depth': 1,	

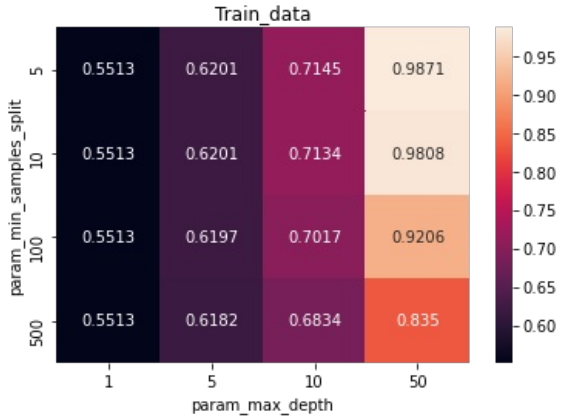
2	2.652350	0.202783	0.033989	0.002933	1	100	'min_samples_split': 100}	0.55
3	2.559553	0.045672	0.029716	0.003347	1	500	{'max_depth': 1, 'min_samples_split': 500}	0.55
4	5.445992	0.344330	0.031442	0.002514	5	5	{'max_depth': 5, 'min_samples_split': 5}	0.60

In [167…
```
train_auc = results['mean_train_score']
test_auc = results['mean_test_score']
max_depth = results['param_max_depth']
sample_split = results['param_min_samples_split']
```

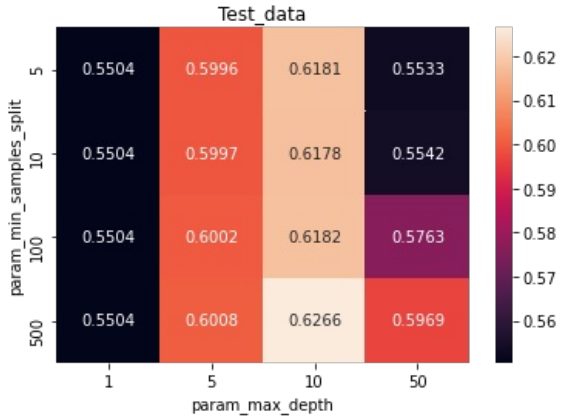
plotting the performance of model both on train data and cross validation data for each hyper parameter(set1)

In [211…
```
# https://www.kaggle.com/paulrohan2020/decision-trees-on-donors-choose/notebook
auc_scores = results.groupby(['param_min_samples_split', 'param_max_depth']).max()
auc_scores = auc_scores.unstack()[['mean_test_score', 'mean_train_score']]
# print(type(auc_scores))
```

In [212…
```
sns.heatmap(auc_scores['mean_train_score'], annot = True, fmt = '.4g')
plt.title('Train_data')

plt.show()
```



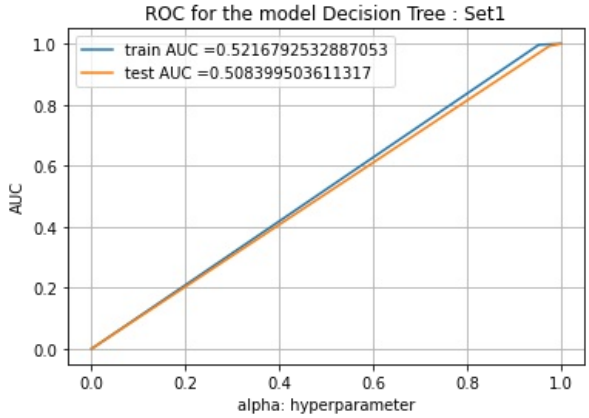
In [214…
```
sns.heatmap(auc_scores['mean_test_score'], annot = True, fmt = '.4g')
plt.title('Test_data')
plt.show()
```



Train the model with best param, and find the AUC on test data and plot the ROC curve on both train and test(Set1)

In [220…
```
# Best Paramters: {'max_depth': 10, 'min_samples_split': 500}
DT_s1 = DecisionTreeClassifier(max_depth = 10, min_samples_split = 500)
DT_s1.fit(X_train_set1, y_train)
y_trn_pred = DT_s1.predict(X_train_set1)
y_tst_pred = DT_s1.predict(X_test_set1)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_trn_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_tst_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
```

```
plt.ylabel("AUC")
plt.title("ROC for the model Decision Tree : Set1")
plt.grid()
plt.show()
```



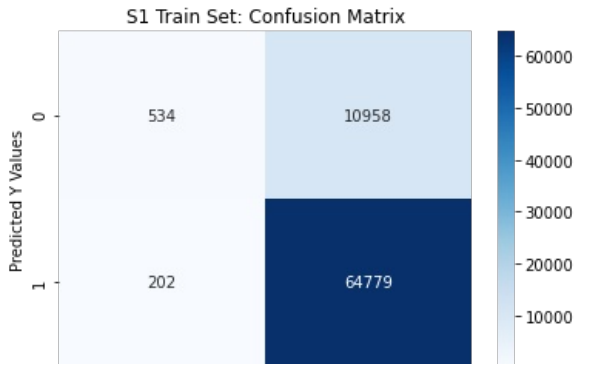
Confusion matrix for set1

In [221…
```
# reference notebook sample solutions from AAIC
# 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
```

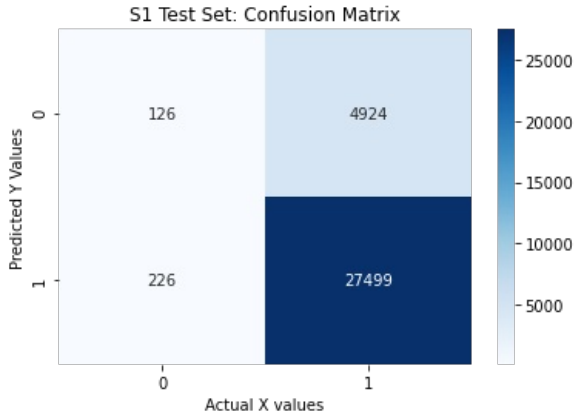
In [223…
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
cm_trn = confusion_matrix(y_train, predict_with_best_t(y_trn_pred, best_t))
print(cm_trn)
print("Test confusion matrix")
cm_test = confusion_matrix(y_test, predict_with_best_t(y_tst_pred, best_t))
print(cm_test)
```

```
the maximum value of tpr*(1-fpr) 0.04632265985720431 for threshold 1
Train confusion matrix
[[  534 10958]
 [  202 64779]]
Test confusion matrix
[[  126  4924]
 [  226 27499]]
```

In [224…
```
# https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-exponenti
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cm_trn, annot=True, fmt="d",cmap='Blues')

plt.title('S1 Train Set: Confusion Matrix')
plt.xlabel('Actual X values')
plt.ylabel('Predicted Y Values')
plt.show()
```



In [225…
```
sns.heatmap(cm_test, annot=True, fmt="d",cmap='Blues')

plt.title('S1 Test Set: Confusion Matrix')
plt.xlabel('Actual X values')
plt.ylabel('Predicted Y Values')
plt.show()
```



In [227…
```
#function to retrieve false positive indices
# https://www.kaggle.com/paulrohan2020/decision-trees-on-donors-choose/notebook
def FP_indices(y_actual, y_pred):
    FP_list = (y_actual == 0) & (y_pred == 1)
    print(FP_list[:10])
    y_val_FP = y_actual[FP_list]
    indices_FP = np.in1d(y_actual, y_val_FP).nonzero()[0]
    return indices_FP
```

Plot the WordCloud with the words of essay text of these `false positive data points` (Set1)

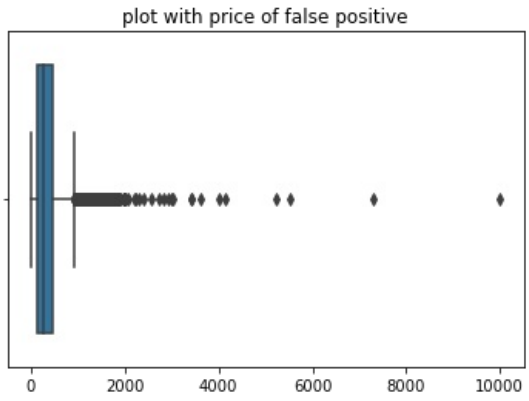
In [251…
```
# https://www.geeksforgeeks.org/generating-word-cloud-python/
def word_plotter(text):
    comment_words = ''
    stopwords = set(STOPWORDS)
    for val in text:

        # 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 = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)
    # plot the WordCloud image
    plt.figure(figsize = (8, 12), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    plt.show()
```

In [ ]:
```
FP_test = FP_indices(y_test, y_tst_pred)
X_test_essay = X_test['essay'].values
essay_FP = X_test_essay[FP_test]
word_plotter(essay_FP)
```

Plot the box plot with the `price` of `false positive data points` (Set1)

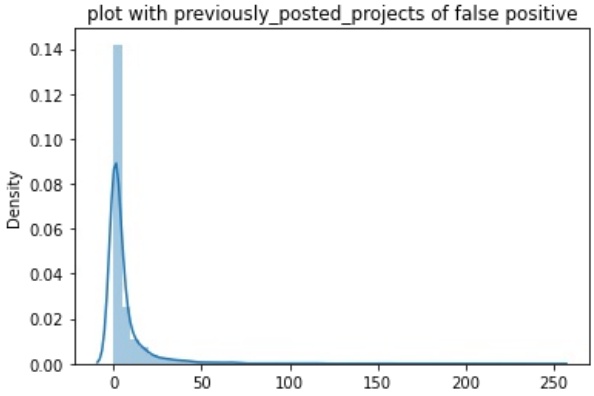
In [302…
```
X_test_price = X_test['price'].values
```

```
price_FP = X_test_price[FP_test]
sns.boxplot(price_FP)
plt.title('plot with price of false positive')
plt.show()
```



Plot the pdf with the `teacher_number_of_previously_posted_projects` of `false positive data points` (Set1)

In [301…
```
X_test_proj = X_test['teacher_number_of_previously_posted_projects'].values
proj_FP = X_test_proj[FP_test]
sns.distplot(proj_FP)
plt.title('plot with previously_posted_projects of false positive')
plt.show()
```



Hyperparameter Tuning Set 2 features with GridSearchCV

In [256…
```
depth = [1, 5, 10, 50]
sample_split =  [5, 10, 100, 500]
params = {'max_depth' : depth, 'min_samples_split' : sample_split}
DT = DecisionTreeClassifier(class_weight = 'balanced')
model_s2 = GridSearchCV(DT, params, cv = 3, scoring = 'roc_auc', return_train_score = True)
model_s2.fit(X_trn_set2, y_train)
print('Best Paramters:', model_s2.best_params_)
```

```
Best Paramters: {'max_depth': 5, 'min_samples_split': 500}
```

In [257…
```
results = pd.DataFrame.from_dict(model_s2.cv_results_)
# results = results.sort_values(['mean_train_score', 'mean_test_score'])
```

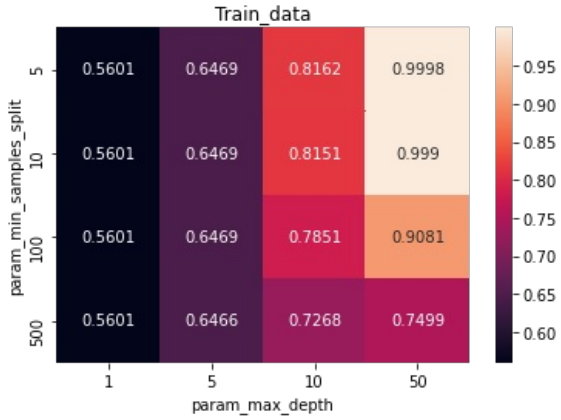
In [258…
```
train_auc = results['mean_train_score']
test_auc = results['mean_test_score']
max_depth = results['param_max_depth']
sample_split = results['param_min_samples_split']
```

plotting the performance of model both on train data and cross validation data for each hyper parameter(Set 2)

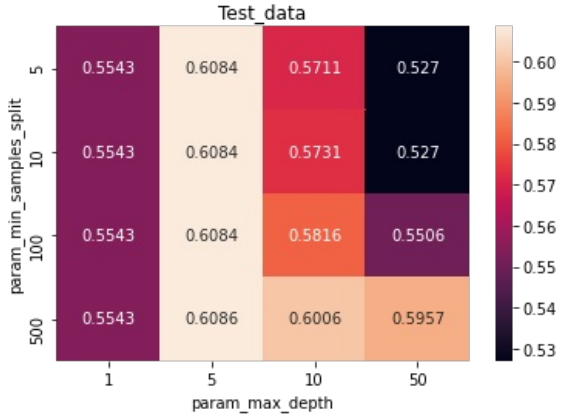
In [259…
```
# https://www.kaggle.com/paulrohan2020/decision-trees-on-donors-choose/notebook
auc_scores = results.groupby(['param_min_samples_split', 'param_max_depth']).max()
auc_scores = auc_scores.unstack()[['mean_test_score', 'mean_train_score']]
# print(type(auc_scores))
```

In [260…
```
sns.heatmap(auc_scores['mean_train_score'], annot = True, fmt = '.4g')
```

```
plt.title('Train_data')

plt.show()
```

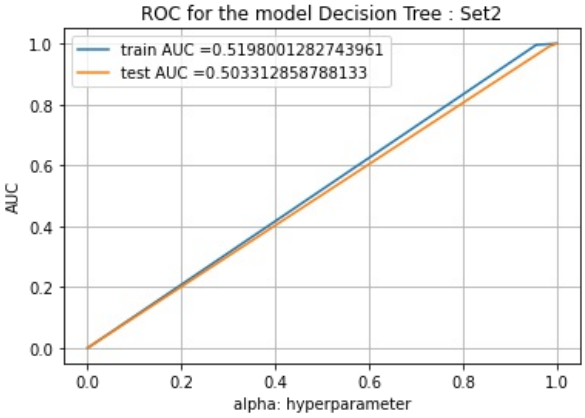


In [261…
```
sns.heatmap(auc_scores['mean_test_score'], annot = True, fmt = '.4g')
plt.title('Test_data')
plt.show()
```



Train the model with best param, and find the AUC on test data and plot the ROC curve on both train and test(Set 2)

In [262…
```
# Best Paramters: {'max_depth': 10, 'min_samples_split': 500}
DT_s2 = DecisionTreeClassifier(max_depth = 10, min_samples_split = 500)
DT_s2.fit(X_trn_set2, y_train)
y_trn_pred = DT_s2.predict(X_trn_set2)
y_tst_pred = DT_s2.predict(X_tst_set2)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_trn_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_tst_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ROC for the model Decision Tree : Set2")
plt.grid()
plt.show()
```



Confusion matrix for Set 2

In [263…
```
# reference notebook sample solutions from AAIC
```

```
# 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
```

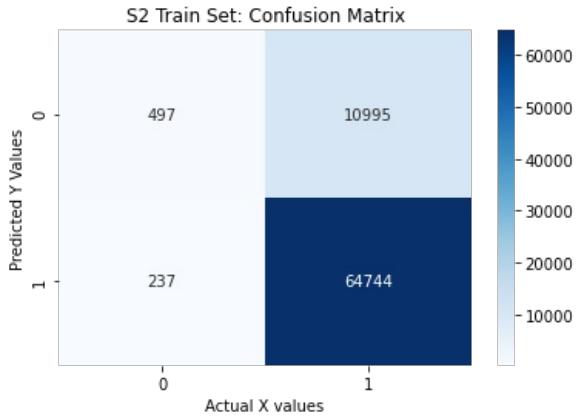
In [264…

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
cm_trn = confusion_matrix(y_train, predict_with_best_t(y_trn_pred, best_t))
print(cm_trn)
print("Test confusion matrix")
cm_test = confusion_matrix(y_test, predict_with_best_t(y_tst_pred, best_t))
print(cm_test)
```

```
the maximum value of tpr*(1-fpr) 0.04308974344601189 for threshold 1
Train confusion matrix
[[  497 10995]
 [  237 64744]]
Test confusion matrix
[[   81  4969]
 [  261 27464]]
```

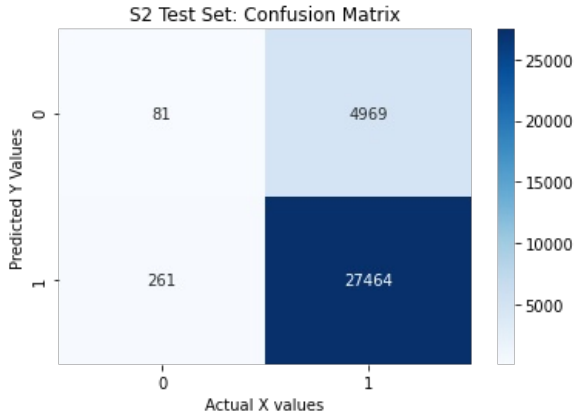
In [265…

```
# https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-exponenti
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cm_trn, annot=True, fmt="d",cmap='Blues')

plt.title('S2 Train Set: Confusion Matrix')
plt.xlabel('Actual X values')
plt.ylabel('Predicted Y Values')
plt.show()
```



In [266…

```
sns.heatmap(cm_test, annot=True, fmt="d",cmap='Blues')

plt.title('S2 Test Set: Confusion Matrix')
plt.xlabel('Actual X values')
plt.ylabel('Predicted Y Values')
plt.show()
```



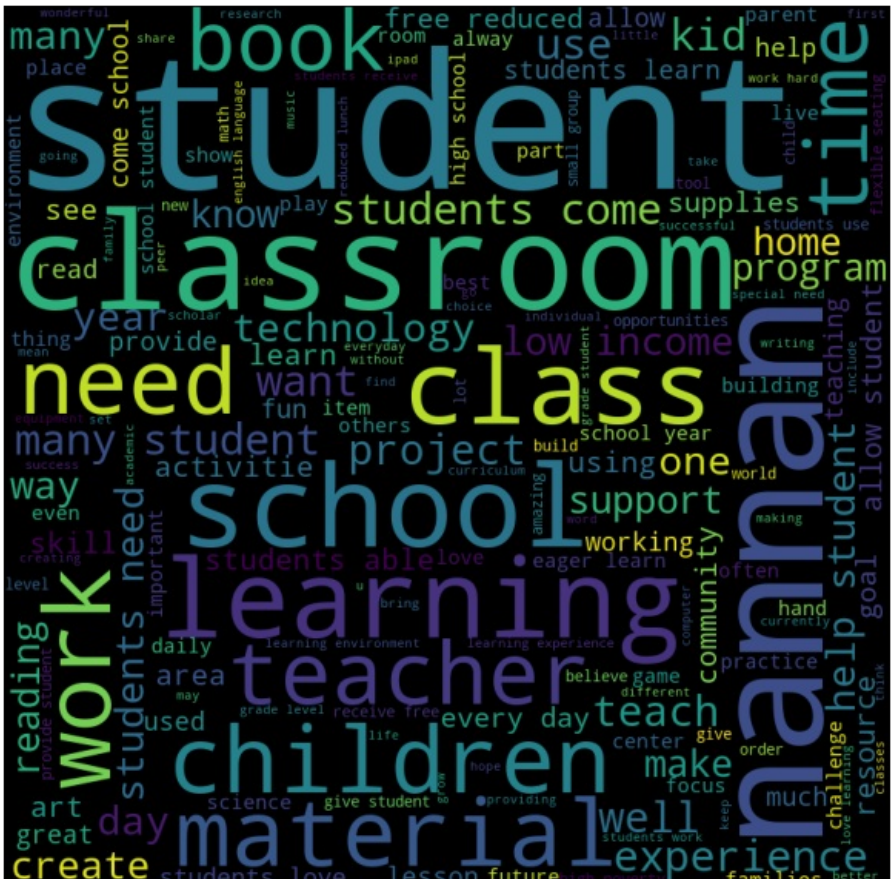
In [267…
```
#function to retrieve false positive indices
# https://www.kaggle.com/paulrohan2020/decision-trees-on-donors-choose/notebook
def FP_indices(y_actual, y_pred):
    FP_list = (y_actual == 0) & (y_pred == 1)
    print(FP_list[:10])
    y_val_FP = y_actual[FP_list]
    indices_FP = np.in1d(y_actual, y_val_FP).nonzero()[0]
    return indices_FP
```

Plot the WordCloud with the words of essay text of these `false positive data points` (Set 2)

In [268…
```
# https://www.geeksforgeeks.org/generating-word-cloud-python/
def word_plotter(text):
    comment_words = ''
    stopwords = set(STOPWORDS)
    for val in text:

        # 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 = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)
    # plot the WordCloud image
    plt.figure(figsize = (8, 12), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    plt.show()
```

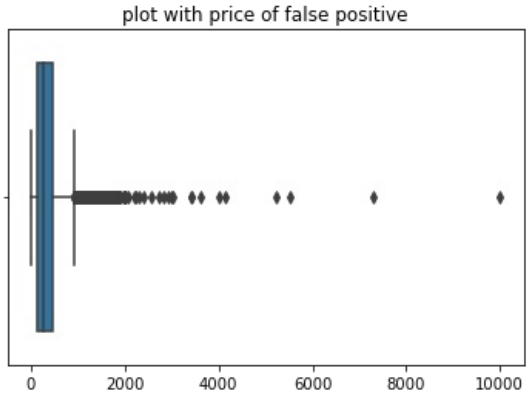
In [269…
```
FP_test = FP_indices(y_test, y_tst_pred)
X_test_essay = X_test['essay'].values
essay_FP = X_test_essay[FP_test]
word_plotter(essay_FP)
```

```
[False False False False False False False  True False False]
```



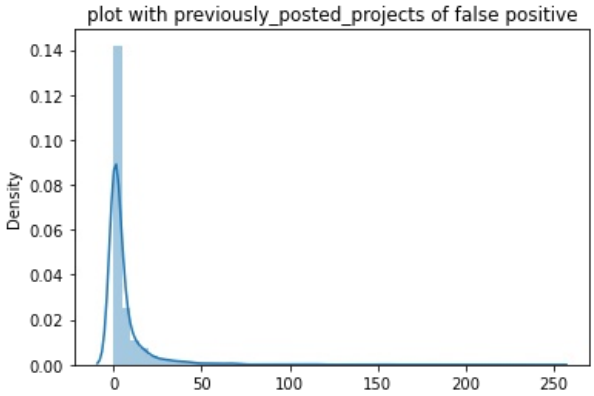
Plot the box plot with the `price` of `false positive data points` (Set 2)

In [303…
```
X_test_price = X_test['price'].values
price_FP = X_test_price[FP_test]
sns.boxplot(price_FP)
plt.title('plot with price of false positive')
plt.show()
```



Plot the pdf with the `teacher_number_of_previously_posted_projects` of `false positive data points` (Set 2)

In [304…
```
X_test_proj = X_test['teacher_number_of_previously_posted_projects'].values
proj_FP = X_test_proj[FP_test]
sns.distplot(proj_FP)
plt.title('plot with previously_posted_projects of false positive')
plt.show()
```



TASK - 2

feature importance using 'feature*importances*' and chosen model is LogisticRegression

In [324…
```
# https://www.kaggle.com/paulrohan2020/decision-trees-on-donors-choose/notebook
X_train_set1 = X_train_set1.tocsr()
X_test_set1 = X_test_set1.tocsr()
clf = DecisionTreeClassifier(class_weight='balanced')
clf.fit(X_train_set1, y_train)
feature_imp = np.array(clf.feature_importances_)
```

In [308…
```
set1_feature_imp_trn = X_train_set1[:, feature_imp > 0]
set1_feature_imp_tst = X_test_set1[:, feature_imp > 0]
log_reg = LogisticRegression(random_state=0)
```

In [309…
```
hyper_param = {'C' : [0, 0.5, 1, 1.5, 2, 10, 50, 100]}
```

HyperParam tuning using gridsearchCV

In [310…
```
gridsearch = GridSearchCV(log_reg, hyper_param, cv = 3)
```

In [311…
```
gridsearch.fit(set1_feature_imp, y_train)
```

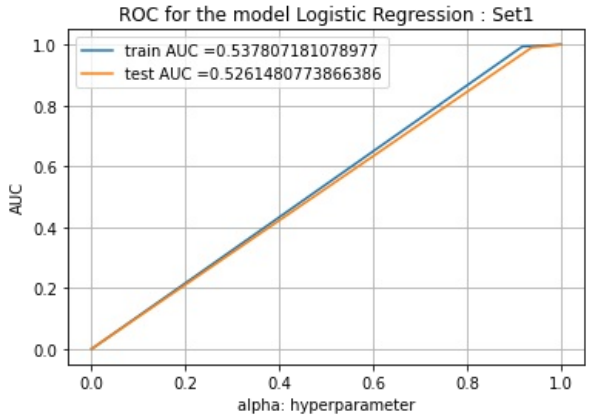
Out[311…
```
GridSearchCV(cv=3, estimator=LogisticRegression(random_state=0),
             param_grid={'C': [0, 0.5, 1, 1.5, 2, 10, 50, 100]})
```

In [312…
```
print("best parameter", gridsearch.best_params_)
```

```
best parameter {'C': 2}
```

In [313…
```
log_reg = LogisticRegression(C = 2).fit(set1_feature_imp, y_train)
```

In [316…
```
#naive bayes assignment
y_trn_pred = log_reg.predict(set1_feature_imp_trn)
y_tst_pred = log_reg.predict(set1_feature_imp_tst)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_trn_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_tst_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ROC for the model Logistic Regression : Set1")
plt.grid()
plt.show()
```



In [317…
```
confusion_matrix(y_train, predict_with_best_t(y_trn_pred, best_t))
```

Out[317…
```
array([[  942, 10550],
       [  413, 64568]], dtype=int64)
```

In [319…
```
confusion_matrix(y_test, predict_with_best_t(y_tst_pred, best_t))
```

Out[319…
```
array([[  312,  4738],
       [  263, 27462]], dtype=int64)
```

Summary of models and vectorizer used with train and test AUC score

In [321…
```
#NaiveBays Assignment
column = ['Vectorizer', 'Model',  'Train_AUC', 'Test_AUC']
row1 = ['Tfidf', 'DecisionTree', 0.52, 0.50]
row2 = ['TfidfW2V', 'DecisionTree', 0.51, 0.50]
row3 = ['Tfidf', 'LogisticRegression', 0.53, 0.52]
```

In [322…
```
summary = pd.DataFrame(data = [row1, row2, row3], columns = column, index = None)
```

In [323…
```
import tabulate as tb
print(tb.tabulate(summary, headers='keys', tablefmt='psql'))
```

```
+----+-------------+--------------------+-------------+------------+
|    | Vectorizer  | Model              |   Train_AUC |   Test_AUC |
|----+-------------+--------------------+-------------+------------|
|  0 | Tfidf       | DecisionTree       |        0.52 |       0.5  |
|  1 | TfidfW2V    | DecisionTree       |        0.51 |       0.5  |
|  2 | Tfidf       | LogisticRegression |        0.53 |       0.52 |
+----+-------------+--------------------+-------------+------------+
```

In [ ]: