Homework 5

Xiaoxu Ruan

a) Data Cleaning

i)

In [1]:

```
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import os
```

In [2]:

```
stack_train = pd.read_csv('stack_stats_2020_train.csv')
stack_test = pd.read_csv('stack_stats_2020_test.csv')
stack_train1 = pd.concat([stack_train, stack_test],axis=0).reset_index()
```

In [ ]:

```

```

ii) convert html texts to plain texts, remove \n

In [3]:

```
def soup_csv(x):
    try:
        a = BeautifulSoup(x, 'html').get_text().strip().replace("\n", " ")
        return a
    except:
        return x

print ('Parsing html in table...')
soup = stack_train1[['Body','Title']].applymap(soup_csv)
```

```
Parsing html in table...
```

In [4]:

```
stack_train1['Body'] = soup['Body']
stack_train1['Title'] = soup['Title']
```

iii) Other Transformation

Remove Latex from "Body"

In [5]:

```
import re
def remove_latex(document):
    for i in range(len(re.findall(r"\$(.*?)\$", document))):
        document = document.replace(re.findall(r"\$(.*?)\$", document)[i], '')
    return document
```

In [6]:

```
stack_train1['Body'] = stack_train1['Body'].apply(remove_latex)
```

In [7]:

```
stack_train1.head(10)
```

Out[7]:

	index	Id	Score	Body	Title	Tags
0	0	495560	1	I have a set of data that I am transforming us...	R: emmeans back tranform clr data using clrInv	<r><mixed-model><linear><lsmeans>
1	1	489896	0	We are sending a one bit message to someone. ...	Trying to determine the failure rate of redund...	<probability><python>
2	2	497951	2	I am aware that there is a similar post: Vecto...	How to derive categorical cross entropy update...	<logistic><cross-entropy>
3	3	478542	2	I have a Poisson distributed glm where I have ...	Learning more about glm parameters, how to dig...	<generalized-linear-model><interpretation>
4	4	458388	0	1) how do i decide which transformation or sca...	Is there I guide to decide which transformatio...	<python><data-transformation><dataset><feature...
5	5	476035	1	Suppose , *where* is a function unknown to ...	What happens to kernel regression (Nadaraya–Wa...	<kernel-smoothing><change-point><derivative><s...
6	6	450570	8	What ort of kernel denity etimator doe one ue ...	Kernel density estimation and boundary bias	<kernel-smoothing><density-estimation><bias-co...
7	7	481773	1	Suppose $$ be a random sample form normal dist...	When will the type 1 and type 2 error be the s...	<hypothesis-testing><self-study><normal-distri...
8	8	490701	2	I found some R code for performing ridge regre...	Ridge regression not working for very simple d...	<r><regression><ridge-regression>
9	9	473268	0	Assume I have a model following ARIMA(p,q,d) w...	Sampling with python statsmodels ARIMA package	<time-series><python><sampling><arima><stochas...

In [8]:

```
def get_formula_count(document):
    number = document.count('$$')
    return number
```

In [9]:

```
stack_train1['formula_count'] = stack_train1['Body'].apply(get_formula_count)
stack_train1[:9]
```

Out[9]:

	index	Id	Score	Body	Title	Tags	formula_count
0	0	495560	1	I have a set of data that I am transforming us...	R: emmeans back tranform clr data using clrInv	<r><mixed-model><linear><lsmeans>	0
1	1	489896	0	We are sending a one bit message to someone. ...	Trying to determine the failure rate of redund...	<probability><python>	0
2	2	497951	2	I am aware that there is a similar post: Vecto...	How to derive categorical cross entropy update...	<logistic><cross-entropy>	0
3	3	478542	2	I have a Poisson distributed glm where I have ...	Learning more about glm parameters, how to dig...	<generalized-linear-model><interpretation>	0
4	4	458388	0	1) how do i decide which transformation or sca...	Is there I guide to decide which transformatio...	<python><data-transformation><dataset><feature...	0
5	5	476035	1	Suppose $, where$ is a function unknown to ...	What happens to kernel regression (Nadaraya–Wa...	<kernel-smoothing><change-point><derivative><s...	22
6	6	450570	8	What ort of kernel denity etimator doe one ue ...	Kernel density estimation and boundary bias	<kernel-smoothing><density-estimation><bias-co...	8
7	7	481773	1	Suppose $$ be a random sample form normal dist...	When will the type 1 and type 2 error be the s...	<hypothesis-testing><self-study><normal-distri...	7
8	8	490701	2	I found some R code for performing ridge regre...	Ridge regression not working for very simple d...	<r><regression><ridge-regression>	1

In [10]:

```
def del_formula(document):
    doc = document.replace('$$', '')
    return doc
```

In [11]:

```
stack_train1['Body'] = stack_train1['Body'].apply(del_formula)
```

Remove Other Language from "Body"

In [12]:

```
def remove_otherlanguage(document):
    clean = re.sub("[^a-zA-Z0-9]+", " ", document)
    return clean
```

In [13]:

```
stack_train1['Body'] = stack_train1['Body'].apply(remove_otherlanguage)
```

Convert into Lowercase

In [14]:

```
text_lowercase_body = stack_train1['Body'].apply(str.lower)
text_lowercase_title = stack_train1['Title'].apply(str.lower)
```

Remove "<",">" in Column: "Tags"

In [15]:

```
train_tags = [item.replace("<","").replace(">"," ") for item in stack_train1['Ta
gs']]
train_tags = [item.split(" ") for item in train_tags]
train_tags = [item[:-1] for item in train_tags]

stack_train1['Tags'] = train_tags
```

In [16]:

```
stack_train1.head(10)
```

Out[16]:

	index	Id	Score	Body	Title	Tags	formula_count
0	0	495560	1	I have a set of data that I am transforming us...	R: emmeans back tranform clr data using clrInv	[r, mixed-model, linear, lsmeans]	0
1	1	489896	0	We are sending a one bit message to someone Th...	Trying to determine the failure rate of redund...	[probability, python]	0
2	2	497951	2	I am aware that there is a similar post Vector...	How to derive categorical cross entropy update...	[logistic, cross-entropy]	0
3	3	478542	2	I have a Poisson distributed glm where I have ...	Learning more about glm parameters, how to dig...	[generalized-linear-model, interpretation]	0
4	4	458388	0	1 how do i decide which transformation or scal...	Is there I guide to decide which transformatio...	[python, data-transformation, dataset, feature...	0
5	5	476035	1	Suppose where is a function unknown to the res...	What happens to kernel regression (Nadaraya–Wa...	[kernel-smoothing, change-point, derivative, s...	22
6	6	450570	8	What ort of kernel denity etimator doe one ue ...	Kernel density estimation and boundary bias	[kernel-smoothing, density-estimation, bias-co...	8
7	7	481773	1	Suppose be a random sample form normal distrib...	When will the type 1 and type 2 error be the s...	[hypothesis-testing, self-study, normal-distri...	7
8	8	490701	2	I found some R code for performing ridge regre...	Ridge regression not working for very simple d...	[r, regression, ridge-regression]	1
9	9	473268	0	Assume I have a model following ARIMA p q d wi...	Sampling with python statsmodels ARIMA package	[time-series, python, sampling, arima, stochas...	0

Remove Punctuation

In [17]:

```
from string import punctuation

def remove_punctuation(document):

    no_punct = ''.join([character for character in document if character not in 
punctuation])

    return no_punct
```

In [18]:

```
text_no_punc_body = text_lowercase_body.apply(remove_punctuation)
text_no_punc_title = text_lowercase_title.apply(remove_punctuation)
```

Remove Digits

In [19]:

```
def remove_digit(document):

    no_digit = ''.join([character for character in document if not character.isd
igit()])

    return no_digit
```

In [20]:

```
text_no_digit_body = text_no_punc_body.apply(remove_digit)
text_no_digit_title = text_no_punc_title.apply(remove_digit)
```

iv) Text Cleaning by nltk

Tokenization

In [21]:

```
import nltk
```

In [22]:

```
# tokenization
from nltk.tokenize import word_tokenize

text_tokenized_body = text_no_digit_body.apply(word_tokenize)
text_tokenized_title = text_no_digit_title.apply(word_tokenize)
```

Remove Stopwords

In [23]:

```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
```

In [24]:

```
def remove_stopwords(document):

    words = [word for word in document if not word in stop_words]

    return words
```

In [25]:

```
text_no_stop_body = text_tokenized_body.apply(remove_stopwords)
text_no_stop_title = text_tokenized_title.apply(remove_stopwords)
```

Stemming

In [26]:

```
from nltk.stem import PorterStemmer

porter = PorterStemmer()

def stemmer(document):

    stemmed_document = [porter.stem(word) for word in document]

    return stemmed_document
```

In [27]:

```
text_stemmed_body = text_no_stop_body.apply(stemmer)
text_stemmed_title = text_no_stop_title.apply(stemmer)
```

In [ ]:

```

```

Detokenization

In [28]:

```
from nltk.tokenize.treebank import TreebankWordDetokenizer

text_detokenized_body = text_stemmed_body.apply(TreebankWordDetokenizer().detoke
nize)
text_detokenized_title = text_stemmed_title.apply(TreebankWordDetokenizer().deto
kenize)
stack_train1['Tags'] = stack_train1['Tags'].apply(TreebankWordDetokenizer().deto
kenize)
```

In [29]:

```
stack_train1['Body'] = text_detokenized_body
stack_train1['Title'] = text_detokenized_title
text_detokenized_tags = stack_train1['Tags']
```

In [30]:

```
stack_train1.head(5)
```

Out[30]:

	index	Id	Score	Body	Title	Tags	formula_count
0	0	495560	1	set data transform use clr function librari co...	r emmean back tranform clr data use clrinv	r mixed-model linear lsmeans	0
1	1	489896	0	send one bit messag someon chanc messag bit tr...	tri determin failur rate redundantli send bit ...	probability python	0
2	2	497951	2	awar similar post vector cross entropi loss lo...	deriv categor cross entropi updat rule multicl...	logistic cross-entropy	0
3	3	478542	2	poisson distribut glm identifi origin paramet ...	learn glm paramet dig deeper	generalized-linear-model interpretation	0
4	4	458388	0	decid transform scale use pass data machin lea...	guid decid transform choos differ scenario typ...	python data-transformation dataset feature-eng...	0

Document-term Matrix

In [31]:

```
from sklearn.feature_extraction.text import CountVectorizer

countvec = CountVectorizer()

sparse_tags = countvec.fit_transform(text_detokenized_tags)
tags_dtm = pd.DataFrame(sparse_tags.toarray(), columns=countvec.get_feature_name
s(),index = stack_train1.index)
sparse_title = countvec.fit_transform(text_detokenized_title)
title_dtm = pd.DataFrame(sparse_title.toarray(), columns=countvec.get_feature_na
mes(),index = stack_train1.index)
sparse_body = countvec.fit_transform(text_detokenized_body)
body_dtm = pd.DataFrame(sparse_body.toarray(), columns=countvec.get_feature_name
s(),index = stack_train1.index)
body_dtm
```

Out[31]:

	aa	aaa	aaabbb	aaabbbc	aaacccc	aaadd	aaai	aab	aaba	aabababbaa	...	zx
0	0	0	0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	0	0	0	0	...	0
2	0	0	0	0	0	0	0	0	0	0	...	0
3	0	0	0	0	0	0	0	0	0	0	...	0
4	0	0	0	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
27491	0	0	0	0	0	0	0	0	0	0	...	0
27492	0	0	0	0	0	0	0	0	0	0	...	0
27493	0	0	0	0	0	0	0	0	0	0	...	0
27494	0	0	0	0	0	0	0	0	0	0	...	0
27495	0	0	0	0	0	0	0	0	0	0	...	0

27496 rows × 49044 columns

v) Dealing three types of data independently -- "Body", "Title", "Tags"

In [32]:

```
body_suffix = body_dtm.add_suffix('_Body')
title_suffix = title_dtm.add_suffix('_Title')
tags_suffix = tags_dtm.add_suffix('_Tags')
```

In [33]:

```
body_suffix.head(10)
```

Out[33]:

	aa_Body	aaa_Body	aaabbb_Body	aaabbbc_Body	aaacccc_Body	aaadd_Body	aaai_Body
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0

10 rows × 49044 columns

In [34]:

```
dtm_all = pd.concat([body_suffix, title_suffix, tags_suffix], axis=1)
```

In [35]:

```
dtm_all.head(10)
```

Out[35]:

	aa_Body	aaa_Body	aaabbb_Body	aaabbbc_Body	aaacccc_Body	aaadd_Body	aaai_Body
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0

10 rows × 60325 columns

b)

Reduce the Column Size

min_df1 = 0.005

In [36]:

```
# 0.5% of the data:

countvec1 = CountVectorizer(min_df=0.005)

sparse_tags1 = countvec1.fit_transform(text_detokenized_tags)
tags_dtm1 = pd.DataFrame(sparse_tags1.toarray(), columns=countvec1.get_feature_n
ames(),index = stack_train1.index)
sparse_title1 = countvec1.fit_transform(text_detokenized_title)
title_dtm1 = pd.DataFrame(sparse_title1.toarray(), columns=countvec1.get_feature
_names(),index = stack_train1.index)
sparse_body1 = countvec1.fit_transform(text_detokenized_body)
body_dtm1 = pd.DataFrame(sparse_body1.toarray(), columns=countvec1.get_feature_n
ames(),index = stack_train1.index)
body_dtm1
```

Out[36]:

	ab	abil	abl	absolut	accept	access	accord	account	accur	accuraci	...	wrong
0	0	0	0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	0	0	0	0	...	0
2	0	0	0	0	0	0	0	0	0	0	...	0
3	0	0	0	0	0	0	0	0	0	0	...	0
4	0	0	0	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
27491	0	0	0	0	0	0	0	0	0	0	...	0
27492	0	0	0	0	1	0	0	0	0	0	...	0
27493	0	0	0	0	0	0	0	0	0	0	...	0
27494	0	0	0	0	0	0	0	0	0	0	...	0
27495	0	0	0	0	0	0	0	1	0	0	...	0

27496 rows × 1424 columns

min_df2 = 0.01

Supervised Learning with Text

In [39]:

```
y_train = (stack_train['Score'] >= 1).astype(int)
y_test = (stack_test['Score'] >= 1).astype(int)

X = pd.concat([body_dtm1, title_dtm1, tags_dtm1], axis = 1)
X['formula_count'] = stack_train1['Body'].apply(get_formula_count)
X_train = X.iloc[:19247]
X_test = X.iloc[19247:].reset_index()
```

In [40]:

```
X_test.head()
```

Out[40]:

	level_0	ab	abil	abl	absolut	accept	access	accord	account	accur	...	time	transfo
0	19247	0	0	0	0	0	0	0	0	0	...	0	
1	19248	0	0	0	0	0	0	0	0	0	...	0	
2	19249	0	0	0	0	0	0	0	0	0	...	0	
3	19250	0	0	0	0	0	0	0	0	0	...	0	
4	19251	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 1844 columns

In [41]:

```
X_test = X_test.drop('level_0', axis = 1)
```

Baseline model

In [42]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

def evaluation(true, pred):
    cm = confusion_matrix(true, pred)
    TPR = cm[1][1]/(cm[1][1] + cm[1][0])
    FPR = cm[0][1]/(cm[0][1] + cm[0][0])
    Acc = accuracy_score(true, pred)
    return Acc, TPR, FPR
```

In [43]:

```
if sum(y_train) / len(y_train) < 0.5:
    y_train0 = np.zeros(len(y_train))
    y_test0 = np.zeros(len(y_test))
else:
    y_train0 = np.ones(len(y_train))
    y_test0 = np.ones(len(y_test))
```

In [44]:

```
acc_train0, TPR_train0, FPR_train0 = evaluation(y_train, y_train0)
acc_test0, TPR_test0, FPR_test0 = evaluation(y_test, y_test0)
print(acc_test0, TPR_test0, FPR_test0)
```

```
0.5123045217602133 0.0 0.0
```

Logistic Regression

In [45]:

```
from sklearn.linear_model import LogisticRegression

logi = LogisticRegression(random_state = 88)
logi.fit(X_train, y_train)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_log
istic.py:765: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[45]:

```
LogisticRegression(random_state=666)
```

In [46]:

```
y_prob_train = logi.predict_proba(X_train)
y_logi_train = pd.Series([1 if x > 0.5 else 0 for x in y_prob_train[:,1]], \
                         index = y_train.index)
y_prob_test = logi.predict_proba(X_test)
y_logi_test = pd.Series([1 if x > 0.5 else 0 for x in y_prob_test[:,1]], \
                        index = y_test.index)
```

In [47]:

```
acc_logi_train, TPR_logi_train, FPR_logi_train = evaluation(y_train, y_logi_trai
n)
acc_logi_test, TPR_logi_test, FPR_logi_test = evaluation(y_test, y_logi_test)
```

CART

In [ ]:

```
# Running grid search to identify the best ccp_alpha
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

grid_values = {'ccp_alpha': np.linspace(0, 0.1, 101)}

dtc = DecisionTreeClassifier(random_state=88)
dtc_cv = GridSearchCV(dtc, param_grid=grid_values, cv=10).fit(X_train, y_train)
```

In [ ]:

```
import matplotlib.pyplot as plt

ccp_alpha = dtc_cv.cv_results_['param_ccp_alpha'].data
ACC_scores = dtc_cv.cv_results_['mean_test_score']

plt.figure(figsize=(8, 6))
plt.xlabel('ccp_alpha', fontsize=16)
plt.ylabel('CV Accuracy', fontsize=16)
plt.scatter(ccp_alpha, ACC_scores, s=3)
plt.plot(ccp_alpha, ACC_scores, linewidth=3)
plt.grid(True, which='both')

plt.tight_layout()
plt.show()

print('Best ccp_alpha', dtc_cv.best_params_)
```

In [ ]:

```
from sklearn.tree import plot_tree

print('Node count =', dtc_cv.best_estimator_.tree_.node_count)
plt.figure(figsize=(6,6))
plot_tree(dtc_cv.best_estimator_,
          feature_names=X_train.columns,
          class_names=['0','1'],
          filled=True,
          impurity=False,
          fontsize=12)
plt.show()
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

y_pred = dtc_cv.predict(X_test)
dtc_cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", dtc_cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

In [ ]:

```
## According to the result of grid search, we apply the best cpp_alpha into the
 CART
```

In [ ]:

```
%%time
dtc = DecisionTreeClassifier(min_samples_leaf = 5,\
                             ccp_alpha = 0.001,\
                             random_state = 88).fit(X_train, y_train)
```

In [ ]:

```
y_cart_train = dtc.predict(X_train)
y_cart_test = dtc.predict(X_test)
```

In [52]:

```
acc_cart_train, TPR_cart_train, FPR_cart_train = evaluation(y_train, y_cart_trai
n)
acc_cart_test, TPR_cart_test, FPR_cart_test = evaluation(y_test, y_cart_test)
print(acc_cart_test, TPR_cart_test, FPR_cart_test)
```

```
0.5632197842162686 0.230673626646781 0.12020823473734027
```

Random Forest

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(max_features=5, min_samples_leaf=5, n_estimators=500
, random_state=88)
rf.fit(X_train, y_train)
```

In [ ]:

```
y_pred = rf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

Random Forest Classifier with CV

In [ ]:

```
import time

grid_values = {'max_features': np.linspace(1,120,120, dtype='int32'),
               'min_samples_leaf': [5],
               'n_estimators': [500],
               'random_state': [88]}

tic = time.time()

rf = RandomForestClassifier()
rf_cv = GridSearchCV(rf, param_grid=grid_values, cv=5)
rf_cv.fit(X_train, y_train)

toc = time.time()

print('time:', round(toc-tic, 2),'s')
```

In [ ]:

```
max_features = rf_cv.cv_results_['param_max_features'].data
ACC_scores = rf_cv.cv_results_['mean_test_score']

plt.figure(figsize=(8, 6))
plt.xlabel('max_features', fontsize=16)
plt.ylabel('CV Accuracy', fontsize=16)
plt.scatter(max_features, ACC_scores, s=3)
plt.plot(max_features, ACC_scores, linewidth=3)
plt.grid(True, which='both')

plt.tight_layout()
plt.show()

print('Best parameters', rf_cv.best_params_)
```

In [ ]:

```
y_pred = rf_cv.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

In [ ]:

```
sorted_idx = rf_cv.best_estimator_.feature_importances_.argsort()

feature_importances = rf_cv.best_estimator_.feature_importances_[sorted_idx[::-1
]]
feature_names = X_train.columns[sorted_idx[::-1]]

plt.figure(figsize=(8,7))
plt.barh(feature_names[:10], 100*feature_importances[:10])
plt.show()
```

Gradient Boosting Classifier

In [ ]:

```
from sklearn.ensemble import GradientBoostingClassifier

grid_values = {'n_estimators': np.linspace(100, 5000, 50, dtype='int32'),  # np.
logspace(6, 12, num=7, base=2, dtype='int32'),
               'learning_rate': [0.01],
               'max_leaf_nodes': np.linspace(2, 10, 8, dtype='int32'),
               'max_depth': [100],
               'min_samples_leaf': [10],
               'random_state': [88]}

tic = time.time()

gbc = GradientBoostingClassifier()
gbc_cv = GridSearchCV(gbc, param_grid=grid_values, cv=5)
gbc_cv.fit(X_train, y_train)

toc = time.time()

print('time:', round(toc-tic, 2),'s')
```

In [ ]:

```
n_estimators = gbc_cv.cv_results_['param_n_estimators'].data
cv_acc_scores = gbc_cv.cv_results_['mean_test_score']

plt.figure(figsize=(12, 8))
plt.xlabel('n estimators', fontsize=16)
plt.ylabel('CV Accuracy', fontsize=16)
plt.grid(True, which='both')

N = len(grid_values['max_leaf_nodes'])
M = len(grid_values['n_estimators'])
for i in range(N):
    plt.scatter(n_estimators[(M*i):(M*i)+M], cv_acc_scores[(M*i):(M*i)+M], s=30)
    plt.plot(n_estimators[(M*i):(M*i)+M], cv_acc_scores[(M*i):(M*i)+M], linewidt
h=2,
             label='max leaf nodes = '+str(grid_values['max_leaf_nodes'][i]))
plt.legend(loc='lower right')
plt.show()
```

In [ ]:

```
y_pred = gbc_cv.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

c)

i) Select a model

I will select a model with the highest **TPR**, so that we can put those questions predicted to be most valuable on the top of the page. Compared with TPR, TFR and accuracy are not so important, since we don't care about the least valuable questions.

ii) Revisit the model

In [ ]:

```

```