HW4 Skeleton Code

Please note that this skeleton code is provided to help you with homework. Full description of each question can be found on HW5.pdf, so please read instruction of each question carefully. There might be some questions that is not presented in this code.

In [2]:
```
import os
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
```

Q. Changing HTML Text to Plain Text

The Python library **BeautifulSoup** is useful for dealing with html text. In order to use this library, you will need to install it first by running the following command: **conda install beautifulsoup4** in the terminal.
In the code, you can import it by running the following line:
from bs4 import BeautifulSoup

In [2]:
```
#conda install beautifulsoup4
```

In [3]:
```
#Read our data file

df_train = pd.read_csv('stack_stats_2023_train.csv') #Todo
df_test = pd.read_csv('stack_stats_2023_test.csv') #Todo
```

In [4]:
```
df_train.head(5)
```

Out[4]:

	Id	Score	Body	Title	Tags
0	502641	1	<p>I'm a master's student in EECS working my w...	Why does the PyTorch tutorial on DQN define st...	<machine-learning><reinforcement-learning><q-l...
1	477291	1	<p>I do not know if this is a good question, b...	Does random walking have a memory?	<probability><law-of-large-numbers>
2	448489	4	<p>I am doing 10 times repeated 10-fold cross-...	Which statistic to report for repeated cross-v...	<cross-validation>
3	487075	0	<p>I have a dataset with 1MM records, around 4...	Binary classification on imbalanced data - odd...	<unbalanced-classes><calibration>
4	481670	2	<p>I want to run a regression where one of the...	How to best summarize Likert data (to use as a...	<multiple-regression><missing-data><likert><it...

In [5]:
```
#Cleaning 'Body'
#Change HTML Text to Plain text using get_text() function from BeautifulSoup
#If you are not familiar with the apply method, please check discussion week 1(

df_train['Body'] = df_train['Body'].apply(lambda text: BeautifulSoup(text,'htm
```

In [6]:
```
#Manually cleaned up newline tag \n and tab tag \t.
df_train['Body'] = df_train['Body'].apply(lambda text: text.replace('\n',''))
df_train['Body'] = df_train['Body'].apply(lambda text: text.replace('\t',''))

#If you need any other cleaning process, please uncomment the below.
#df_train['Body'] = df_train['Body'].apply(lambda ) #Todo

#Cleaning Tags
#This would be somewhat similar to the above.

#Todo: Clean Tags, please feel free to add any lines below
df_train['Tags'] = df_train['Tags'].apply(lambda text: text.replace('>',''))
df_train['Tags'] = df_train['Tags'].apply(lambda text: text.replace('<',''))


#Todo: Repeat the same process for test dataset

df_test['Body'] = df_test['Body'].apply(lambda text: BeautifulSoup(text,'html.p
df_test['Body'] = df_test['Body'].apply(lambda text: text.replace('\n',''))
df_test['Body'] = df_test['Body'].apply(lambda text: text.replace('\t',''))
#Cleaning Tags
df_test['Tags'] = df_test['Tags'].apply(lambda text: text.replace('>',''))
df_test['Tags'] = df_test['Tags'].apply(lambda text: text.replace('<',''))
```

In [6]:
```
df_train.head(5)
```

Out[6]:

	Id	Score	Body	Title	Tags
0	502641	1	I'm a master's student in EECS working my way ...	Why does the PyTorch tutorial on DQN define st...	machine-learningreinforcement-learningq-learning
1	477291	1	I do not know if this is a good question, but ...	Does random walking have a memory?	probabilitylaw-of-large-numbers
2	448489	4	I am doing 10 times repeated 10-fold cross-val...	Which statistic to report for repeated cross-v...	cross-validation
3	487075	0	I have a dataset with 1MM records, around 40 f...	Binary classification on imbalanced data - odd...	unbalanced-classescalibration
4	481670	2	I want to run a regression where one of the ex...	How to best summarize Likert data (to use as a...	multiple-regressionmissing-datalikertitem-resp...

Q. Basic Text Cleaning and Merging into a single Text data

Change to Lower Case, Remove puncuation, digits,

In [7]:
```
#Change to Lowercase

df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(s
```

In [8]:
```
df_train
```

Out[8]:

	Id	Score	Body	Title	Tags
0	502641	1	i'm a master's student in eecs working my way ...	why does the pytorch tutorial on dqn define st...	machine-learningreinforcement-learningq-learning
1	477291	1	i do not know if this is a good question, but ...	does random walking have a memory?	probabilitylaw-of-large-numbers
2	448489	4	i am doing 10 times repeated 10-fold cross-val...	which statistic to report for repeated cross-v...	cross-validation
3	487075	0	i have a dataset with 1mm records, around 40 f...	binary classification on imbalanced data - odd...	unbalanced-classescalibration
4	481670	2	i want to run a regression where one of the ex...	how to best summarize likert data (to use as a...	multiple-regressionmissing-datalikertitem-resp...
...	...	...	...	...	...
19242	458552	1	i need to fill missing values. i have found th...	is matrix factorization also going to work wit...	machine-learningdata-imputationrecommender-sys...
19243	486912	6	in the vast majority of cases, linear regressi...	in reality, there is almost always measurement...	regressionmodelingmeasurement-errorerrors-in-v...
19244	489944	1	i can see on the wikipedia page of the poisson...	slight difference in the pmf of the poisson di...	poisson-distribution
19245	493843	1	there are three conditions to prove that a fun...	how to prove that a function is 2-increasing (...	machine-learningmathematical-statisticscumulat...
19246	447244	2	i have a timecourse rnaseq experiment for muta...	how to test if this there is a genotypic effec...	rbioinformatics

19247 rows × 5 columns

In [9]:

```
#df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].apply(s
#df_train
```

In [8]:

```
#Remove Punctations
from string import punctuation

#You can get this function from our discussion session code. However, we leave
def remove_punctuation(document):
```

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

    return no_punct
```

In [9]:
```
df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(r
```

In [10]:
```
#Remove Digits

def remove_digit(document):

    no_digit = ''.join([char for char in document if not char.isdigit()])

    return no_digit

df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(r
```

In [13]:
```
df_train
```

Out[13]:

	Id	Score	Body	Title	
0	502641	1	im a masters student in eecs working my way to...	why does the pytorch tutorial on dqn define st...	machinelearningreinforcementlearningqle
1	477291	1	i do not know if this is a good question but i...	does random walking have a memory	probabilitylawoflargenu
2	448489	4	i am doing times repeated fold crossvalidatio...	which statistic to report for repeated crossva...	crossval
3	487075	0	i have a dataset with mm records around featu...	binary classification on imbalanced data odd ...	unbalancedclassescali
4	481670	2	i want to run a regression where one of the ex...	how to best summarize likert data to use as an...	multipleregressionmissingdatalikertitemres
...	...	...	...	...	
19242	458552	1	i need to fill missing values i have found tha...	is matrix factorization also going to work wit...	machinelearningdataimputationrecommendersy
19243	486912	6	in the vast majority of cases linear regressio...	in reality there is almost always measurement ...	regressionmodelingmeasurementerrorerrorsi
19244	489944	1	i can see on the wikipedia page of the poisson...	slight difference in the pmf of the poisson di...	poissondistr
19245	493843	1	there are three conditions to prove that a fun...	how to prove that a function is increasing copula	machinelearningmathematicalstatisticscumu
19246	447244	2	i have a timecourse rnaseq experiment for muta...	how to test if this there is a genotypic effec...	rbioinfor

19247 rows × 5 columns

Tokenization and Remove Stopwords and do stemming

In [11]:
```
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(w
```

```
[nltk_data] Downloading package punkt to /Users/yitinggan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [15]:
```
df_train.head()
```

Out[15]:

	Id	Score	Body	Title	Tags
0	502641	1	[im, a, masters, student, in, eecs, working, m...	[why, does, the, pytorch, tutorial, on, dqn, d...	[machinelearningreinforcementlearningqlearning]
1	477291	1	[i, do, not, know, if, this, is, a, good, ques...	[does, random, walking, have, a, memory]	[probabilitylawoflargenumbers]
2	448489	4	[i, am, doing, times, repeated, fold, crossval...	[which, statistic, to, report, for, repeated, ...	[crossvalidation]
3	487075	0	[i, have, a, dataset, with, mm, records, aroun...	[binary, classification, on, imbalanced, data,...	[unbalancedclassescalibration]
4	481670	2	[i, want, to, run, a, regression, where, one, ...	[how, to, best, summarize, likert, data, to, u...	[multipleregressionmissingdatalikertitemrespon...

In [12]:
```
#Remove Stopwords

from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def remove_stopwords(document):

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

    return words

df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(r
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/yitinggan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [17]:
```
df_train.head()
```

Out[17]:

	Id	Score	Body	Title	Tags
0	502641	1	[im, masters, student, eecs, working, way, tow...	[pytorch, tutorial, dqn, define, state, differ...	[machinelearningreinforcementlearningqlearning]
1	477291	1	[know, good, question, found, answer, anywhere...	[random, walking, memory]	[probabilitylawoflargenumbers]
2	448489	4	[times, repeated, fold, crossvalidation, want,...	[statistic, report, repeated, crossvalidation]	[crossvalidation]
3	487075	0	[dataset, mm, records, around, features, class...	[binary, classification, imbalanced, data, odd...	[unbalancedclassescalibration]
4	481670	2	[want, run, regression, one, explanatory, vari...	[best, summarize, likert, data, use, independe...	[multipleregressionmissingdatalikertitemrespon...

In [13]:
```
#We use porter stemming

from nltk.stem import PorterStemmer

porter = PorterStemmer()

def stemmer(document):

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

    return stemmed_document

df_train[['Body','Title','Tags']] = df_train[['Body','Title','Tags']].applymap
df_test[['Body','Title','Tags']] = df_test[['Body','Title','Tags']].applymap(s
```

Let's Check our dataframe

In [19]:
```
df_train.head(5)
```

Out[19]:

	Id	Score	Body	Title	Tags
0	502641	1	[im, master, student, eec, work, way, toward, ...	[pytorch, tutori, dqn, defin, state, differ]	[machinelearningreinforcementlearningqlearn]
1	477291	1	[know, good, question, found, answer, anywher,...	[random, walk, memori]	[probabilitylawoflargenumb]
2	448489	4	[time, repeat, fold, crossvalid, want, report,...	[statist, report, repeat, crossvalid]	[crossvalid]
3	487075	0	[dataset, mm, record, around, featur, class, i...	[binari, classif, imbalanc, data, odd, calibr,...	[unbalancedclassescalibr]
4	481670	2	[want, run, regress, one, explanatori, variabl...	[best, summar, likert, data, use, independ, va...	[multipleregressionmissingdatalikertitemrespon...

Q. Treat Three text data independently and merge into one column

In [14]:

```
#Treat Three types of data independently
#let's define functions that will help this operation

def add_body(document):

    added_document = [word+'body' for word in document]

    return added_document

def add_title(document):

    added_document = [word+'title' for word in document]

    return added_document

def add_tags(document):

    added_document = [word+'tags' for word in document]

    return added_document
```

In [15]:

```
df_train['Body'] = df_train['Body'].apply(add_body)
df_train['Title'] = df_train['Title'].apply(add_title)
df_train['Tags'] = df_train['Tags'].apply(add_tags)
```

```
df_test['Body'] = df_test['Body'].apply(add_body)
df_test['Title'] = df_test['Title'].apply(add_title)
df_test['Tags'] = df_test['Tags'].apply(add_tags)
```

In [16]:
```
#Now we need to merge all those 3 columns into a single column. Implement this
df_train['text'] = df_train['Body'] + df_train['Title'] + df_train['Tags']
df_test['text'] = df_test['Body'] + df_test['Title'] + df_test['Tags']
```

Let's check our DataFrame

In [23]:
```
df_train.head(5)
```

Out[23]:

	Id	Score	Body	Title	Tags
0	502641	1	[imbody, masterbody, studentbody, eecbody, wor...	[pytorchtitle, tutorititle, dqntitle, defintit...	[machinelearningreinforcementlearningqlearntags]
1	477291	1	[knowbody, goodbody, questionbody, foundbody, ...	[randomtitle, walktitle, memorititle]	[probabilitylawoflargenumbtags]
2	448489	4	[timebody, repeatbody, foldbody, crossvalidbod...	[statisttitle, reporttitle, repeattitle, cross...	[crossvalidtags]
3	487075	0	[datasetbody, mmbody, recordbody, aroundbody, ...	[binarititle, classiftitle, imbalanctitle, dat...	[unbalancedclassescalibrtags]
4	481670	2	[wantbody, runbody, regressbody, onebody, expl...	[besttitle, summartitle, likerttitle, datatitl...	[multipleregressionmissingdatalikertitemrespon...

Q. Detokenize and convert to document term matrices

In [17]:
```
#Merge Three text column into one column and detokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer
from sklearn.feature_extraction.text import CountVectorizer

text_train = df_train['text'].apply(TreebankWordDetokenizer().detokenize) #Todo
countvec_train = CountVectorizer(min_df = 0.01)
sparse_dtm_train = countvec_train.fit_transform(text_train)
```

In [18]:
```
#Todo: Do same on the test set.
text_test = df_test['text'].apply(TreebankWordDetokenizer().detokenize)
sparse_dtm_test = countvec_train.transform(text_test)
```

In [19]:
```
#Convert the sprase dtm to pandas DataFrame.
dtm_train = pd.DataFrame(data=sparse_dtm_train.toarray(),index=df_train.index,
```

```
dtm_test = pd.DataFrame(data=sparse_dtm_test.toarray(),index=df_test.index, co
```

Q. Change dependent variable to binary variable

In [20]:
```
#Change 'Score' to a binary variable, which indicates whether the question is g
y_train = (df_train['Score']>=1).astype(int)
y_test = (df_test['Score']>=1).astype(int)
```

In [21]:
```
#Add y_train and y_test to your data frame if it is needed. Drop unnecessary co
df_train['GoodQuestion'] = y_train
df_test['GoodQuestion'] = y_test
df_train.drop(columns = ['Score','Id'], inplace = True)
df_test.drop(columns = ['Score','Id'], inplace = True)
```

Let's check our DataFrame

In [29]:
```
df_train.head(5)
```

Out[29]:

	Body	Title	Tags	text
0	[imbody, masterbody, studentbody, eecbody, wor...	[pytorchtitle, tutorititle, dqntitle, defintit...	[machinelearningreinforcementlearningqlearntags]	[imbody, masterbody, studentbody, eecbody, wor...
1	[knowbody, goodbody, questionbody, foundbody, ...	[randomtitle, walktitle, memorititle]	[probabilitylawoflargenumbtags]	[knowbody, goodbody, questionbody, foundbody, ...
2	[timebody, repeatbody, foldbody, crossvalidbod...	[statisttitle, reporttitle, repeattitle, cross...	[crossvalidtags]	[timebody, repeatbody, foldbody, crossvalidbod...
3	[datasetbody, mmbody, recordbody, aroundbody, ...	[binarititle, classiftitle, imbalanctitle, dat...	[unbalancedclassescalibrtags]	[datasetbody, mmbody, recordbody, aroundbody, ...
4	[wantbody, runbody, regressbody, onebody, expl...	[besttitle, summartitle, likerttitle, datatitl...	[multipleregressionmissingdatalikertitemrespon...	[wantbody, runbody, regressbody, onebody, expl...

(b) Please read the instruction carefully in the pdf.

1.Logistic Regression

In [22]:
```
import statsmodels.api as sm
logreg = sm.Logit(y_train,dtm_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.642065
         Iterations 7
```

In [23]:
```
#statistics
from sklearn.metrics import confusion_matrix
default_false = np.sum(y_train==0)
default_true = np.sum(y_train==1)
print(pd.Series({'0': default_false, '1': default_true}))
```

```
0    9780
1    9467
dtype: int64
```

In [24]:
```
#Baseline Model
baseline_acc = default_false/(default_true+default_false)
baseline_TPR = 0
baseline_FPR = 0
baseline_PRE = 0

#Logistic Regression Model Statistics
log_prob = logreg.predict(dtm_test)
log_pred = pd.Series([1 if x > 0.5 else 0 for x in log_prob], index=log_prob.in
cm = confusion_matrix(y_test, log_pred)
print ("Confusion Matrix : \n", cm)
log_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
log_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
log_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
log_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
```

```
Confusion Matrix :
 [[2398 1732]
 [1927 2192]]
```

2.Linear Discriminant Analysis

In [25]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(dtm_train, y_train)

#LDA Statistics
y_pred = lda.predict(dtm_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
lda_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
lda_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
lda_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
lda_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
```

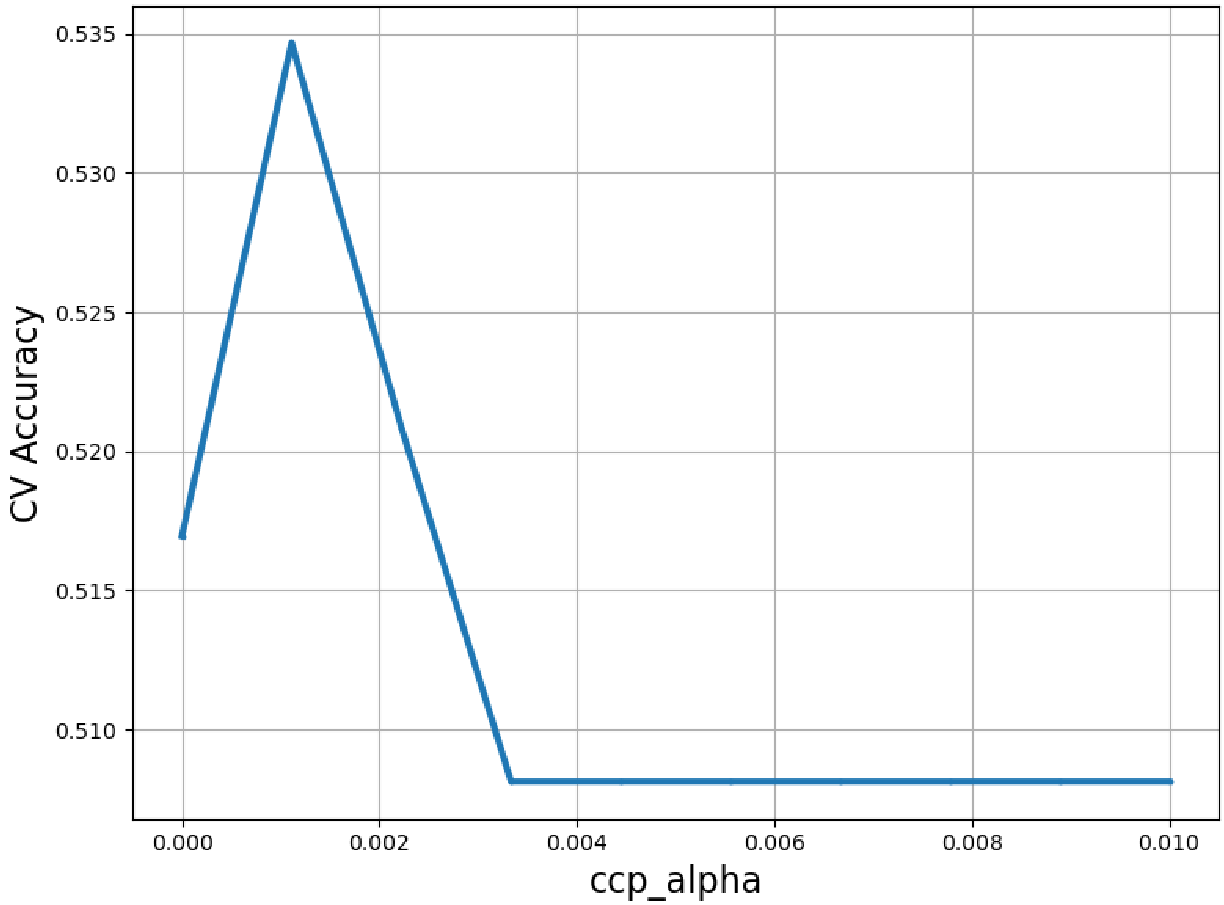
```
Confusion Matrix:
 [[2526 1604]
 [2012 2107]]
```

3.Decision Tree Classifier

In [26]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
grid_values = {'ccp_alpha': np.linspace(0, 0.01, 10)}
dtc = DecisionTreeClassifier(random_state=88)
dtc_cv = GridSearchCV(dtc, param_grid=grid_values, cv=10).fit(dtm_train, y_tra
```

In [27]:
```
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('ccp_alpha', dtc_cv.best_params_)
```




```
ccp_alpha {'ccp_alpha': 0.0011111111111111111}
```

In [28]:
```
#Decision Tree Classifier Statistics
dtc_pred = dtc_cv.best_estimator_.predict(dtm_test)
cm = confusion_matrix(y_test, dtc_pred)
print ("Confusion Matrix : \n", cm)
dtc_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
dtc_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
dtc_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
dtc_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
```

```
Confusion Matrix :
 [[3450  680]
 [3245  874]]
```

4.Random Forest Classifier

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
import time
grid_values = {'max_features': np.linspace(20,40,5, dtype='int32'),
               'min_samples_leaf': [5],
```

```
                'n_estimators': [300],
                'random_state': [88]}
tic = time.time()
rf = RandomForestClassifier()
rf_cv = GridSearchCV(rf, param_grid=grid_values, cv=5)
rf_cv.fit(dtm_train, y_train)
toc = time.time()
print('time:', round(toc-tic, 2),'s')
```

```
time: 1013.34 s
```

In [30]:

```
#Random Forest Classifier Statistics
y_pred = rf_cv.best_estimator_.predict(dtm_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
rf_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
rf_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
rf_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
rf_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
```

```
Confusion Matrix:
 [[2588 1542]
 [1932 2187]]
```

In [31]:

```
#Create Comparison Table
#These lines are provided for you to help construct a comparison table.
#It is not requred to follow this format. + You need to find ACC, TPR, FPR, PRI
comparison_data = {'Baseline':[baseline_acc,baseline_TPR,baseline_FPR, baselin
                    'Logistic Regression':[log_acc,log_TPR,log_FPR, log_PRE],
                    'Decision Tree Classifier':[dtc_acc,dtc_TPR,dtc_FPR,dtc_PRE
                    'Random Forest with CV':[rf_acc,rf_TPR, rf_FPR,rf_PRE],
                   'Linear Discriminant Analysis':[lda_acc,lda_TPR, lda_FPR,lda_

comparison_table = pd.DataFrame(data=comparison_data, index=['Accuracy', 'TPR'
comparison_table.style.set_properties(**{'font-size': '12pt',}).set_table_styl
comparison_table
```

Out[31]:

	Accuracy	TPR	FPR	PRE
Baseline	0.508131	0.000000	0.000000	0.000000
Logistic Regression	0.556431	0.532168	0.419370	0.558614
Decision Tree Classifier	0.524185	0.212187	0.164649	0.562420
Random Forest with CV	0.578858	0.530954	0.373366	0.586484
Linear Discriminant Analysis	0.561644	0.511532	0.388378	0.567771

Answer： I select logistic model, linear discriminant analysis, decision tree classifier, and random forest models. For decision tree classifer and random forest model, 10 fold cross validation with four different parameter values is applied. For logistic model and linear discriminant analysis, we have binary value as dependent variable and the input text as independent variables. Based on the comparison table, I would pick logistic model as my final model because it has a relatively high accuracy and decent TPR, which will be a essential factor to consider in next part.

Report details of your training procedures and final comparisons on the test set in this cell. Use your best judgment to choose a final model and explain your choice.

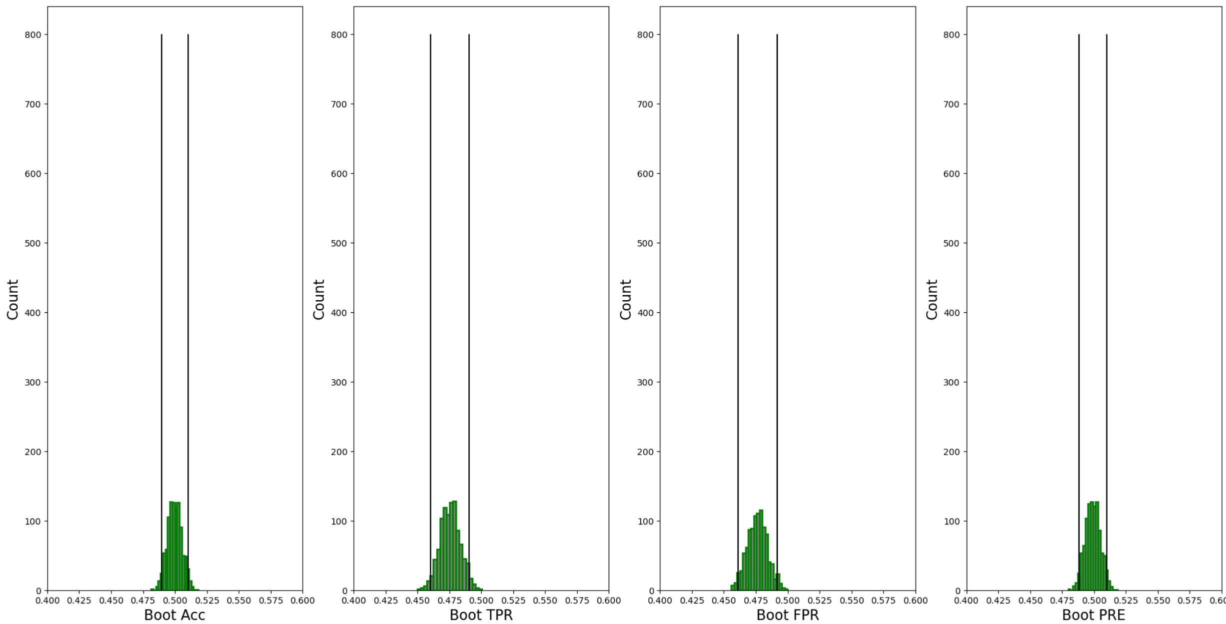
Report Bootstrap Analysis in this cell

In [32]:
```
def bootstrap_validation_logreg(test_data, test_label, model, sample=500,randor
    tic = time.time()
    n_sample = sample
    output_array=np.zeros([n_sample, 4])
    output_array[:]=np.nan
    print(output_array.shape)
    for bs_iter in range(n_sample):
        bs_index = np.random.choice(test_data.index, len(test_data.index),repla
        bs_data = test_data.loc[bs_index]
        bs_label = test_label.loc[bs_index]
        bs_prob = model.predict(bs_data)
        bs_pred = pd.Series([1 if x > 0.5 else 0 for x in bs_prob], index=bs_p
        cm = confusion_matrix(test_label, bs_pred)
        log_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
        log_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
        log_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
        log_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
        output_array[bs_iter,:]=np.array([log_acc,log_TPR,log_FPR,log_PRE])
    output_df = pd.DataFrame(output_array)
    return output_df

bs_output = bootstrap_validation_logreg(dtm_test, y_test, logreg,sample = 1000
CI_acc = np.quantile(bs_output.iloc[:,0],np.array([0.025,0.975]))
CI_TPR = np.quantile(bs_output.iloc[:,1],np.array([0.025,0.975]))
CI_FPR = np.quantile(bs_output.iloc[:,2],np.array([0.025,0.975]))
CI_PRE = np.quantile(bs_output.iloc[:,3],np.array([0.025,0.975]))
mean_acc = np.mean(bs_output.iloc[:,0])
std_acc = np.std(bs_output.iloc[:,0])
mean_TPR = np.mean(bs_output.iloc[:,1])
std_TPR = np.std(bs_output.iloc[:,1])
mean_FPR = np.mean(bs_output.iloc[:,2])
std_FPR = np.std(bs_output.iloc[:,2])
mean_PRE = np.mean(bs_output.iloc[:,3])
std_PRE = np.std(bs_output.iloc[:,3])
fig, axs = plt.subplots(ncols=4, figsize=(24,12))
#Plot Accuracy
axs[0].set_xlabel('Boot Acc', fontsize=16)
axs[0].set_ylabel('Count', fontsize=16)
axs[0].hist(bs_output.iloc[:,0], bins=20,edgecolor='green', linewidth=2,color =
axs[0].set_xlim([0.4,0.6])
axs[0].vlines(x=CI_acc[0], ymin = 0, ymax =800, color = "black")
axs[0].vlines(x=CI_acc[1], ymin = 0, ymax =800, color = "black")
#Plot TPR
axs[1].set_xlabel('Boot TPR', fontsize=16)
axs[1].set_ylabel('Count', fontsize=16)
axs[1].hist(bs_output.iloc[:,1], bins=20,edgecolor='green', linewidth=2,color =
axs[1].set_xlim([0.4,0.6])
axs[1].vlines(x=CI_TPR[0], ymin = 0, ymax =800, color = "black")
```

```
axs[1].vlines(x=CI_TPR[1], ymin = 0, ymax =800, color = "black")
 #Plot FPR
axs[2].set_xlabel('Boot FPR', fontsize=16)
axs[2].set_ylabel('Count', fontsize=16)
axs[2].hist(bs_output.iloc[:,2], bins=20,edgecolor='green', linewidth=2,color =
axs[2].set_xlim([0.4,0.6])
axs[2].vlines(x=CI_FPR[0], ymin = 0, ymax =800, color = "black")
axs[2].vlines(x=CI_FPR[1], ymin = 0, ymax =800, color = "black")
#Plot Precision
axs[3].set_xlabel('Boot PRE', fontsize=16)
axs[3].set_ylabel('Count', fontsize=16)
axs[3].hist(bs_output.iloc[:,3], bins=20,edgecolor='green', linewidth=2,color =
axs[3].set_xlim([0.4,0.6])
axs[3].vlines(x=CI_PRE[0], ymin = 0, ymax =800, color = "black")
axs[3].vlines(x=CI_PRE[1], ymin = 0, ymax =800, color = "black")
plt.show()
```

(1000, 4)



In [33]:
```
#list data
bootstrap_data = {'Accuracy':[CI_acc[0],CI_acc[1],mean_acc,std_acc],'TPR':[CI_
bootstrap_table = pd.DataFrame(data = bootstrap_data, index = ['0.025 quantile
bootstrap_table.style.set_properties(**{'font-size': '12pt',}).set_table_style
bootstrap_table
```

Out[33]:

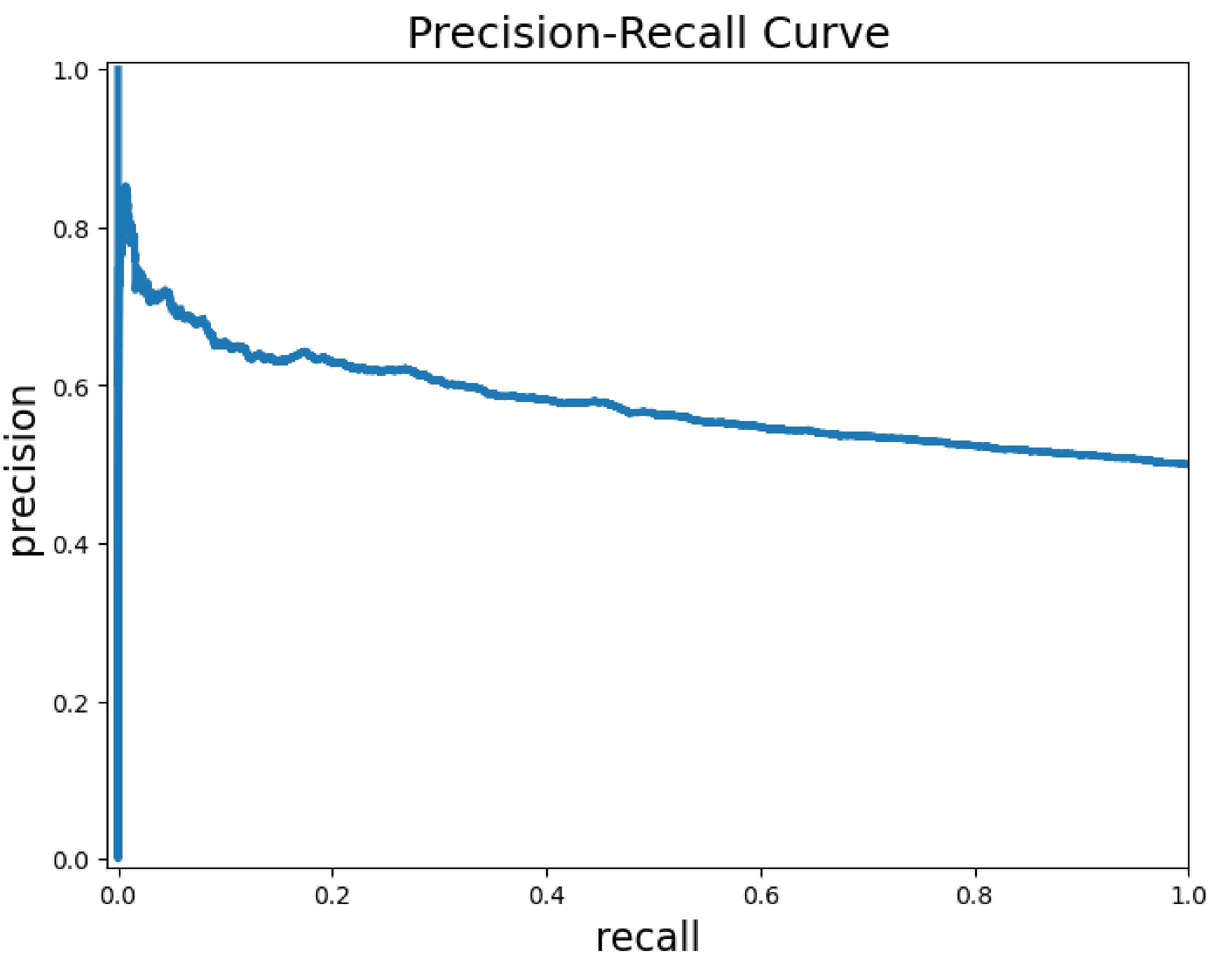
	0.025 quantile	0.975 quantile	Mean	Standard Deviation
Accuracy	0.489390	0.510122	0.499680	0.005486
TPR	0.460063	0.490653	0.475441	0.007859
FPR	0.461017	0.492010	0.476146	0.007720
PRE	0.488144	0.510017	0.498962	0.005765

Answer: As we observed from statistics, the difference between the bootstrap model and the random forest model is enlarged.

(c)

i)I would select a model with the highest precision score. Since our goal is to maximize the probability that the top question is useful, it is important to make our positive prediction as accurate as possible. Precision actually measures the probability of correct detection of positive values. We can see that random forest with cv model has the highest precision score. However, since it takes so long to retrain it we use logistic regression model instead. Logistic regression model also has high accuracy, TPR and precision score.

In [34]:
```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc
#Precision recall curve
precision, recall, _ = precision_recall_curve(y_test,log_prob)
plt.figure(figsize=(8, 6))
plt.title('Precision-Recall Curve', fontsize=18)
plt.xlabel('recall', fontsize=16)
plt.ylabel('precision', fontsize=16)
plt.xlim([-0.01, 1.00])
plt.ylim([-0.01, 1.01])
plt.plot(recall, precision, lw=3)
plt.show()
```



In [35]:
```
temp = recall[1:]>= 0.15
np.nonzero(temp)
```

Out[35]:
```
(array([   0,    1,    2, ..., 7266, 7267, 7268]),)
```

In [36]:
```
p = _[7301]
print(p)
```

```
0.6920363910391362
```

In [37]:
```
log_pred = pd.Series([1 if x > p else 0 for x in log_prob], index=log_prob.inde
cm = confusion_matrix(y_test, log_pred)
print ("Confusion Matrix : \n", cm)
log_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
log_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
log_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
log_PRE = cm.ravel()[3]/(cm.ravel()[1]+cm.ravel()[3])
print('log_acc:',log_acc)
print('log_TPR:',log_TPR)
print('log_FPR:',log_FPR)
print('log_PRE:',log_PRE)
```

```
Confusion Matrix :
 [[3782  348]
 [3520  599]]
log_acc: 0.5310946781428052
log_TPR: 0.1454236465161447
log_FPR: 0.08426150121065375
log_PRE: 0.6325237592397043
```

ii) Based on the graph, we should decrease recall score as much as we can by controlling our threshold parameter. However, notice that we should at least satisfy PR >= 1/7.5 = 0.133333. So, let's find a threshold value that makes TPR close to 0.15. (We are giving some safe buffer to the PR.) Observing the below result, threshold = 0.69 would make our logistic regression model better in precision score and actually it did. Our precision has changed from 0.53 to 0.63 and also FPR has descreased.