INDENG 242 HW2 Ankit Nitinkumar Bhawsar 3038588568

Problem 1

Import Necessary Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

from bs4 import BeautifulSoup
```

Import Training and Testing Dataset and Print Head

In [2]:
```
stack_train = pd.read_csv('stack_stats_2020_train.csv')
stack_test = pd.read_csv('stack_stats_2020_test.csv')
stack_train
```

Out[2]:

	Id	Score	Body	Title	Tags
0	495560	1	<p>I have a set of data that I am transforming...	R: emmeans back tranform clr data using clrInv	<r><mixed-model><linear><lsmeans>
1	489896	0	<p>We are sending a one bit message to someone...	Trying to determine the failure rate of redund...	<probability><python>
2	497951	2	<p>I am aware that there is a similar post: <a...	How to derive categorical cross entropy update...	<logistic><cross-entropy>
3	478542	2	<p>I have a Poisson distributed glm where I ha...	Learning more about glm parameters, how to dig...	<generalized-linear-model><interpretation>
4	458388	0	<p>1) how do i decide which transformation or ...	Is there I guide to decide which transformatio...	<python><data-transformation><dataset><feature...
...	...	...	...	...	...
19242	464995	0	<p>I'm currently trying to implement a decisio...	How does decision tree classify tuple whose co...	<classification><cart>
19243	477516	0	<p>What are the available tools (results) that...	What are the available tools (results) that ca...	<convergence><asymptotics><central-limit-theor...
19244	461894	1	<p>Let $X^n=(X_1,...	Maximum Likelihood Estimator for Censored Data	<estimation><maximum-likelihood><censoring>
19245	498148	0	<p>I have a data set of property sales where i...	Can you use a single missingness indicator for...	<missing-data><data-preprocessing><indicator-f...
19246	469634	0	<p>I have data from a human subject experiment...	What is the appropriate non-parametric test fo...	<r><mixed-model><repeated-measures><nonparamet...

19247 rows × 5 columns

In [3]:
```
print(stack_train.shape,stack_test.shape)
```

```
(19247, 5) (8249, 5)
```

First, we convert the score attribute into Useful attribute where a question (an instance of data) is useful (value is 1) when Score is >=1

In [4]:
```
stack_train['Useful'] = (stack_train['Score'] >= 1).astype('int32').astype('object')
stack_train.drop(columns=['Score'], inplace=True)
stack_train.head()

stack_test['Useful'] = (stack_test['Score'] >= 1).astype('int32').astype('object')
stack_test.drop(columns=['Score'], inplace=True)
stack_test
```

Out[4]:

	Id	Body	Title	Tags	Useful
0	476132	Summarize the problem \...	What I do with the results extracted from lasso?	<stata><lasso>	0
1	450811	I try to calculate the marginal likelihood ...	the marginal likelihood of analytical result i...	<sampling><marginal-distribution><rstan>	0
2	472876	I have a data set which involves 30 binomia...	GLMER Overdispersion and Error messages	<lme4-nlme><glmm><eigenvalues><overdispersion>	1
3	470799	It is well known that the K-means algorithm...	K-medoids: Is there any constraint about the c...	<clustering><k-medoids>	1
4	446472	I'm working on a text classification proble...	Combining XGBoost and LightGBM	<python><boosting>	1
...	...	...	...	...	...
8244	450701	My data is of the form <span class="math-co...	Visualising high dimensional data	<r><data-visualization><ggplot2>	0
8245	481194	I noticed the term ANOVA used in many conte...	Is the analysis of residual variance still ANO...	<regression><anova><generalized-linear-model><...	1
8246	492163	I'm trying to do logistic regression, but I...	Handling missing data in logistic regression	<r><regression><logistic><missing-data><regres...	1
8247	451444	Consider the following experimental design ...	Mixed models: How to treat random factors that...	<r><mixed-model><lme4-nlme>	1
8248	460905	I am constructing different configurations ...	Data partitioning for spatial data	<machine-learning><random-forest><spatial><par...	1

8249 rows × 5 columns

a) Data Preprocessing

Function to remove puntuations and digits

In [5]:
```
def remove_punctuation(document):

    no_punct = ''.join([character for character in document if character not in punctu

    return no_punct

def remove_digit(document):

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

    return no_digit
```

Function to parse body and title columns using BeautifulSoup

In [6]:
```
def html_parse(string_example):

    soup = BeautifulSoup(string_example, 'html.parser')
    string_example = soup.get_text()
    string_example = string_example.replace('\n',' ')
    string_example = string_example.lower()
    string_example = remove_punctuation(string_example)### Function to remove puntuati
    string_example = remove_digit(string_example)
    return string_example
```

Function to parse tags column

In [7]:
```
def tag_parse(string_example):

    string_example = string_example.replace('>',' ')
    string_example = string_example.replace('<','')
    string_example = string_example.lower()
    string_example = remove_punctuation(string_example)
    string_example = remove_digit(string_example)
    return string_example
```

In [8]:
```
body = stack_train['Body']
body = body.apply(html_parse)
```

In [9]:
```
title = stack_train['Title']
title = title.apply(html_parse)
```

In [10]:
```
tags_column = stack_train['Tags']
tags_column = tags_column.apply(tag_parse)
```

Newly parsed data will look like this:

In [11]:
```
stack_train['Body'] = body
stack_train['Title'] = title
stack_train['Tags'] = tags_column

stack_train### Function to remove puntuations and digits
```

Out[11]:

	Id	Body	Title	Tags	Useful
0	495560	i have a set of data that i am transforming us...	r emmeans back tranform clr data using clrinv	r mixedmodel linear lsmeans	1
1	489896	we are sending a one bit message to someone t...	trying to determine the failure rate of redund...	probability python	0
2	497951	i am aware that there is a similar post vector...	how to derive categorical cross entropy update...	logistic crossentropy	1
3	478542	i have a poisson distributed glm where i have ...	learning more about glm parameters how to dig ...	generalizedlinearmodel interpretation	1
4	458388	how do i decide which transformation or scali...	is there i guide to decide which transformatio...	python datatransformation dataset featureengin...	0
...	...	...	...	...	...
19242	464995	im currently trying to implement a decision tr...	how does decision tree classify tuple whose co...	classification cart	0
19243	477516	what are the available tools results that can ...	what are the available tools results that can ...	convergence asymptotics centrallimittheorem es...	0
19244	461894	let xnxxxn denote a sample where ximathbf ep...	maximum likelihood estimator for censored data	estimation maximumlikelihood censoring	1
19245	498148	i have a data set of property sales where info...	can you use a single missingness indicator for...	missingdata datapreprocessing indicatorfunction	0
19246	469634	i have data from a human subject experiment wi...	what is the appropriate nonparametric test for...	r mixedmodel repeatedmeasures nonparametric wi...	0

19247 rows × 5 columns

Tokenize the columns data for further processing

In [12]:
```
import nltk
nltk.download('punkt')
```

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

Out[12]:
```
True
```

In [13]:
```
from nltk.tokenize import word_tokenize

body_tokens = stack_train['Body'].apply(word_tokenize)
title_tokens = stack_train['Title'].apply(word_tokenize)
tags_column_tokens = stack_train['Tags'].apply(word_tokenize)

print(body_tokens.head(10))
```

```
0    [i, have, a, set, of, data, that, i, am, trans...
1    [we, are, sending, a, one, bit, message, to, s...
2    [i, am, aware, that, there, is, a, similar, po...
3    [i, have, a, poisson, distributed, glm, where,...
4    [how, do, i, decide, which, transformation, or...
5    [suppose, yigxiei, where, gcdot, is, a, functi...
6    [what, sort, of, kernel, density, estimator, d...
7    [suppose, x, xxn, be, a, random, sample, form,...
8    [i, found, some, r, code, for, performing, rid...
9    [assume, i, have, a, model, following, arimapq...
Name: Body, dtype: object
```

In [14]:
```
print("Average Length of Body Column",np.mean(body_tokens.str.len()))
print("Average Length of Title Column",np.mean(title_tokens.str.len()))
print("Average Length of Tags Column",np.mean(tags_column_tokens.str.len()))
```

```
Average Length of Body Column 155.18065153010858
Average Length of Title Column 9.587987738348833
Average Length of Tags Column 3.228295318750974
```

Remove Stop Words

In [15]:
```
# nltk.download()
```

In [16]:
```
def remove_stopwords(document):

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

    return words
```

In [17]:
```
from nltk.corpus import stopwords

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

body_nostop = body_tokens.apply(remove_stopwords)
title_nostop = title_tokens.apply(remove_stopwords)
tags_column_nostop = tags_column_tokens.apply(remove_stopwords)

print(body_nostop.head(10))
```

```
0    [set, data, transforming, using, clr, function...
1    [sending, one, bit, message, someone, chance, ...
2    [aware, similar, post, vectorization, cross, e...
3    [poisson, distributed, glm, identified, origin...
4    [decide, transformation, scaling, use, passing...
5    [suppose, yigxiei, gcdot, function, unknown, r...
6    [sort, kernel, density, estimator, one, use, a...
7    [suppose, x, xxn, random, sample, form, normal...
8    [found, r, code, performing, ridge, regression...
9    [assume, model, following, arimapqd, statsmode...
Name: Body, dtype: object
```

In [118…
```
print("Average Length of Body Column after removing stopwords: ",np.mean(body_nostop.s
print("Average Length of Title Column after removing stopwords: ",np.mean(title_nostop
print("Average Length of Tags Column after removing stopwords: ",np.mean(tags_column_r
```

```
Average Length of Body Column after removing stopwords:  87.16340208863718
Average Length of Title Column after removing stopwords:  6.221489063230633
Average Length of Tags Column after removing stopwords:  3.228243362601964
```

Stemming the columns data

In [19]:
```
from nltk.stem import PorterStemmer
porter = PorterStemmer()

def stemmer(document):
    stemmed_document = [porter.stem(word) for word in document]
    return stemmed_document
```

In [20]:
```
body_stemmed = body_nostop.apply(stemmer)
title_stemmed = title_nostop.apply(stemmer)
tags_column_stemmed = tags_column_nostop.apply(stemmer)

print(body_stemmed.head(10))
```

```
0    [set, data, transform, use, clr, function, lib...
1    [send, one, bit, messag, someon, chanc, messag...
2    [awar, similar, post, vector, cross, entropi, ...
3    [poisson, distribut, glm, identifi, origin, pa...
4    [decid, transform, scale, use, pass, data, mac...
5    [suppos, yigxiei, gcdot, function, unknown, re...
6    [sort, kernel, densiti, estim, one, use, avoid...
7    [suppos, x, xxn, random, sampl, form, normal, ...
8    [found, r, code, perform, ridg, regress, bosto...
9    [assum, model, follow, arimapqd, statsmodel, p...
Name: Body, dtype: object
```

Creating the Docuent-Term-Matrix

To create the document term matrix, first we need to detokenize the columns since the built-in function for creating DTMs takes strings as inputs

Detokenization

In [21]:
```
from nltk.tokenize.treebank import TreebankWordDetokenizer

body_detokenized = body_stemmed.apply(TreebankWordDetokenizer().detokenize)
title_detokenized = title_stemmed.apply(TreebankWordDetokenizer().detokenize)
tags_column_detokenized = tags_column_stemmed.apply(TreebankWordDetokenizer().detokeni


print(body_detokenized.head(10))
```

```
0    set data transform use clr function librarycom...
1    send one bit messag someon chanc messag bit tr...
2    awar similar post vector cross entropi loss lo...
3    poisson distribut glm identifi origin paramet ...
4    decid transform scale use pass data machin lea...
5    suppos yigxiei gcdot function unknown research...
6    sort kernel densiti estim one use avoid bounda...
7    suppos x xxn random sampl form normal distribu...
8    found r code perform ridg regress bostonh data...
9    assum model follow arimapqd statsmodel packag ...
Name: Body, dtype: object
```

Remove Non-Ascii Characters

Remove any non-ascii characters from the strings. These may include characters in different language scripts and formulae.

In [22]:
```
def remove_non_ascii(string):
    return ''.join(char for char in string if ord(char) < 128)
```

In [23]:
```
body = body_detokenized.apply(remove_non_ascii)
title = title_detokenized.apply(remove_non_ascii)
tags_column = tags_column_detokenized.apply(remove_non_ascii)

print(body.head(10))
```

```
0    set data transform use clr function librarycom...
1    send one bit messag someon chanc messag bit tr...
2    awar similar post vector cross entropi loss lo...
3    poisson distribut glm identifi origin paramet ...
4    decid transform scale use pass data machin lea...
5    suppos yigxiei gcdot function unknown research...
6    sort kernel densiti estim one use avoid bounda...
7    suppos x xxn random sampl form normal distribu...
8    found r code perform ridg regress bostonh data...
9    assum model follow arimapqd statsmodel packag ...
Name: Body, dtype: object
```

Document-term Matrix

In [24]:
```
from sklearn.feature_extraction.text import CountVectorizer
countvec = CountVectorizer()
```

In [25]:
```
body_sparse_dtm = countvec.fit_transform(body)
body_dtm = pd.DataFrame(body_sparse_dtm.toarray(), columns=countvec.get_feature_names(

title_sparse_dtm = countvec.fit_transform(title)
title_dtm = pd.DataFrame(title_sparse_dtm.toarray(), columns=countvec.get_feature_name

tags_column_sparse_dtm = countvec.fit_transform(tags_column)
tags_dtm = pd.DataFrame(tags_column_sparse_dtm.toarray(), columns=countvec.get_feature
```

In [27]:
```
print("Shape of Body Column= ",body_dtm.shape)
print("Shape of Title Column= ",title_dtm.shape)
print("Shape of Tags Column= ",tags_dtm.shape)
```

```
Shape of Body Column=  (19247, 88302)
Shape of Title Column=  (19247, 8122)
Shape of Tags Column=  (19247, 1311)
```

In [119…
```
frequencies = body_dtm.sum().sort_values(ascending=False)
print(frequencies[frequencies > 500])
```

```
model        22386
use          20998
data         19021
variabl      13668
would        11586
             ...
recommend      505
risk           505
percentag      504
iv             503
choic          501
Length: 601, dtype: int64
```

We currently have way too many words, which will make it hard to train our models and may even lead to overfitting. Our solution to the possibility of overfitting is to only keep terms that appear in x% or more of the questions.

We can use the `min_df` parameter, where passing a `float` between `[0.0, 1.0]` will filter according to proportion of appearances in the documents, and passing an `int` will filter according to absolute count.

Removing low frequency words

In [29]:
```
countvec2_body = CountVectorizer(min_df=0.005)

body_sparse_dtm2 = countvec2_body.fit_transform(body)
body_dtm2 = pd.DataFrame(body_sparse_dtm2.toarray(), columns=countvec2_body.get_featur
```

In [30]:
```
countvec2_title = CountVectorizer(min_df=0.005)

title_sparse_dtm2 = countvec2_title.fit_transform(title)
title_dtm2 = pd.DataFrame(title_sparse_dtm2.toarray(), columns=countvec2_title.get_fea
```

In [31]:
```
countvec2_tags = CountVectorizer(min_df=0.005)

tags_column_sparse_dtm2 = countvec2_tags.fit_transform(tags_column)
tags_dtm2 = pd.DataFrame(tags_column_sparse_dtm2.toarray(), columns=countvec2_tags.get
```

In [32]:
```
print("Shape of Body Column= ",body_dtm2.shape)
print("Shape of Title Column= ",title_dtm2.shape)
print("Shape of Tags Column= ",tags_dtm2.shape)
```

```
Shape of Body Column=  (19247, 1393)
Shape of Title Column=  (19247, 226)
Shape of Tags Column=  (19247, 127)
```

Combining to form Dataframe

To differentiate the three differrent columns (body, title, and tags) we add suffixes to the columns titles.

In [33]:
```
body_dtm3 = body_dtm2.add_suffix('_body')
title_dtm3 = title_dtm2.add_suffix('_title')
tags_dtm3 = tags_dtm2.add_suffix('_tags')
```

Final Processed Datasets

As a last step we need to combine the results of preprocessing the three columns into a single dataframe.

In [34]:
```
X_train = pd.concat([body_dtm3, title_dtm3,tags_dtm3], axis=1, join='inner')
X_train.shape
```

Out[34]:
```
(19247, 1746)
```

In [35]:
```
X_train
```

Out[35]:

	ab_body	abil_body	abl_body	absolut_body	accept_body	access_body	accord_body	accoun
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	
...	...	...	...	...	...	...	...	
19242	0	0	2	0	0	0	0	
19243	0	0	0	0	0	0	0	
19244	0	0	0	0	0	0	1	
19245	0	0	0	0	0	0	0	
19246	0	0	0	0	0	0	0	

19247 rows × 1746 columns

In [36]:
```
y_train = stack_train['Useful']
y_train = y_train.astype('int')
y_train
```

Out[36]:
```
0        1
1        0
2        1
3        1
4        0
        ..
19242    0
19243    0
19244    1
19245    0
19246    0
Name: Useful, Length: 19247, dtype: int32
```

Entire Data Cleaning Process for Test Set

Here we repeat the exact same procedure for preprocessing the testing dataset as we did for the training dataset.

In [37]:
```
body_test = stack_test['Body']
body_test = body_test.apply(html_parse)

title_test = stack_test['Title']
title_test = title_test.apply(html_parse)

tags_column_test = stack_test['Tags']
tags_column_test = tags_column_test.apply(tag_parse)

stack_test['Body'] = body_test
stack_test['Title'] = title_test
stack_test['Tags'] = tags_column_test

stack_test
```

Out[37]:

	Id	Body	Title	Tags	Useful
0	476132	summarize the problem i have a dataset with pa...	what i do with the results extracted from lasso	stata lasso	0
1	450811	i try to calculate the marginal likelihood of ...	the marginal likelihood of analytical result i...	sampling marginaldistribution rstan	0
2	472876	i have a data set which involves binomial abs...	glmer overdispersion and error messages	lmenlme glmm eigenvalues overdispersion	1
3	470799	it is well known that the kmeans algorithm is ...	kmedoids is there any constraint about the cho...	clustering kmedoids	1
4	446472	im working on a text classification problem an...	combining xgboost and lightgbm	python boosting	1
...	...	...	...	...	...
8244	450701	my data is of the form xyinmathbbrtimesmathbbr...	visualising high dimensional data	r datavisualization ggplot	0
8245	481194	i noticed the term anova used in many contexts...	is the analysis of residual variance still ano...	regression anova generalizedlinearmodel modeli...	1
8246	492163	im trying to do logistic regression but i cant...	handling missing data in logistic regression	r regression logistic missingdata regressionst...	1
8247	451444	consider the following experimental design wit...	mixed models how to treat random factors that ...	r mixedmodel lmenlme	1
8248	460905	i am constructing different configurations of ...	data partitioning for spatial data	machinelearning randomforest spatial partition...	1

8249 rows × 5 columns

In [38]:

```
body_tokens_test = stack_test['Body'].apply(word_tokenize)
title_tokens_test = stack_test['Title'].apply(word_tokenize)
tags_column_tokens_test = stack_test['Tags'].apply(word_tokenize)

body_nostop_test = body_tokens_test.apply(remove_stopwords)
title_nostop_test = title_tokens_test.apply(remove_stopwords)
tags_column_nostop_test = tags_column_tokens_test.apply(remove_stopwords)

body_stemmed_test = body_nostop_test.apply(stemmer)
title_stemmed_test = title_nostop_test.apply(stemmer)
tags_column_stemmed_test = tags_column_nostop_test.apply(stemmer)

body_detokenized_test = body_stemmed_test.apply(TreebankWordDetokenizer().detokenize)
title_detokenized_test = title_stemmed_test.apply(TreebankWordDetokenizer().detokenize
tags_column_detokenized_test = tags_column_stemmed_test.apply(TreebankWordDetokenizer(

body_test = body_detokenized_test.apply(remove_non_ascii)
title_test = title_detokenized_test.apply(remove_non_ascii)
tags_column_test = tags_column_detokenized_test.apply(remove_non_ascii)
```

```
print(body_test)
```

```
0       summar problem dataset panel data tri make var...
1       tri calcul margin likelihood exampl articl tut...
2       data set involv binomi absencepres total ratio...
3       well known kmean algorithm well design euclide...
4       im work text classif problem compar lightgbm x...
                            ...
8244    data form xyinmathbbrtimesmathbbr denot compon...
8245    notic term anova use mani context one taught a...
8246    im tri logist regress cant seem get result wan...
8247    consid follow experiment design withinsubject ...
8248    construct differ configur random forest order ...
Name: Body, Length: 8249, dtype: object
```

Create Testing Document Term Matrices

Here, the only slight difference is that we use the same CountVectorizer object we used to create the training set so that we don't get different columns for training and testing dataset.

In [39]:
```
body_sparse_dtm2_test = countvec2_body.transform(body_test)
body_dtm2_test = pd.DataFrame(body_sparse_dtm2_test.toarray(), columns=countvec2_body.

title_sparse_dtm2_test = countvec2_title.transform(title_test)
title_dtm2_test = pd.DataFrame(title_sparse_dtm2_test.toarray(), columns=countvec2_tit

tags_column_sparse_dtm2_test = countvec2_tags.transform(tags_column_test)
tags_dtm2_test = pd.DataFrame(tags_column_sparse_dtm2_test.toarray(), columns=countvec


print("Shape of Body Column= ",body_dtm2_test.shape)
print("Shape of Title Column= ",title_dtm2_test.shape)
print("Shape of Tags Column= ",tags_dtm2_test.shape)
```

```
Shape of Body Column=  (8249, 1393)
Shape of Title Column=  (8249, 226)
Shape of Tags Column=  (8249, 127)
```

We can check if all the columns in the training and testing set match or not

In [40]:
```
print("Are all columns same in Body Attribute in Train and Test Set: ",all(body_dtm2_t
print("Are all columns same in Title Attribute in Train and Test Set: ",all(title_dtm2
print("Are all columns same in Tags Attribute in Train and Test Set: ",all(tags_dtm2_t
```

```
Are all columns same in Body Attribute in Train and Test Set:  True
Are all columns same in Title Attribute in Train and Test Set:  True
Are all columns same in Tags Attribute in Train and Test Set:  True
```

In [41]:
```
body_dtm3_test = body_dtm2_test.add_suffix('_body')
title_dtm3_test = title_dtm2_test.add_suffix('_title')
tags_dtm3_test = tags_dtm2_test.add_suffix('_tags')
```

Combine columns to form the testing dataframe

In [42]:
```
X_test = pd.concat([body_dtm3_test, title_dtm3_test,tags_dtm3_test], axis=1, join='inr
X_test.shape
```

Out[42]: (8249, 1746)

In [43]:
```
X_test
```

Out[43]:

	ab_body	abil_body	abl_body	absolut_body	accept_body	access_body	accord_body	account
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	
...	...	...	...	...	...	...	...	
8244	0	0	0	0	0	0	0	
8245	6	0	0	0	1	0	0	
8246	0	0	0	0	0	0	0	
8247	0	0	0	0	0	0	0	
8248	0	0	0	0	0	0	0	

8249 rows × 1746 columns

In [44]:
```
y_test = stack_test['Useful']
y_test = y_test.astype('int')
y_test
```

Out[44]:
```
0       0
1       0
2       1
3       1
4       1
       ..
8244    0
8245    1
8246    1
8247    1
8248    1
Name: Useful, Length: 8249, dtype: int32
```

SUPERVISED LEARNING WITH TEXT

We build models with Logistic Regression, LDA, and CART. We then select the best model for which we use bootstrapping procedure to judge the variability of the results.

In [45]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
```

In [100…
```
def FPR(y_true,y_pred):
    cm = confusion_matrix(y_true, y_pred)
    return cm[0][1]/(cm[0][1]+cm[0][0])
```

In [46]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
print("Are all columns same in Train and Test Set: ",all(X_train.columns == X_test.col
```

```
(19247, 1746)
(19247,)
(8249, 1746)
(8249,)
Are all columns same in Train and Test Set:  True
```

In [47]:
```
print(y_train.value_counts())
print(y_test.value_counts())
```

```
0    9684
1    9563
Name: Useful, dtype: int64
0    4226
1    4023
Name: Useful, dtype: int64
```

In [48]:
```
# Baseline accuracy
print('Baseline Accuracy=', 4226/(8249))
```

```
Baseline Accuracy= 0.5123045217602133
```

Logistic Regression

In [101…
```
from sklearn.linear_model import LogisticRegression

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

Out[101]:
```
LogisticRegression(random_state=88)
```

In [120…
```
y_prob = logreg.predict_proba(X_train)
y_pred = pd.Series([1 if x > 0.5 else 0 for x in y_prob[:,1]], index=y_train.index)

cm = confusion_matrix(y_train, y_pred)
print ("Training Confusion Matrix: \n", cm)
print ("\nTraining Accuracy:", accuracy_score(y_train, y_pred))
print ("Training TPR:", recall_score(y_train, y_pred))
print ("Training FPR:", FPR(y_train, y_pred))
```

```
Training Confusion Matrix:
 [[6722 2962]
 [3541 6022]]

Training Accuracy: 0.6621291629864394
Training TPR: 0.6297187075185611
Training FPR: 0.3058653448988021
```

In [121…
```
y_prob = logreg.predict_proba(X_test)
y_pred = pd.Series([1 if x > 0.5 else 0 for x in y_prob[:,1]], index=y_test.index)
```

```
cm = confusion_matrix(y_test, y_pred)
print ("Testing Confusion Matrix: \n", cm)
print ("\nTesting Accuracy:", accuracy_score(y_test, y_pred))
print ("Testing TPR:", recall_score(y_test, y_pred))
print ("Testing FPR:", FPR(y_test, y_pred))
```

```
Testing Confusion Matrix:
 [[2519 1707]
 [1898 2125]]

Testing Accuracy: 0.5629773305855256
Testing TPR: 0.5282127765349242
Testing FPR: 0.4039280643634643
```

Linear Discriminant Analysis

In [104…
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

Out[104]:
```
LinearDiscriminantAnalysis()
```

In [122…
```
y_pred = lda.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print ("Training Confusion Matrix: \n", cm)
print ("\nTraining Accuracy:", accuracy_score(y_train, y_pred))
print ("Training TPR:", recall_score(y_train, y_pred))
print ("Training FPR:", FPR(y_train, y_pred))
```

```
Training Confusion Matrix:
 [[6748 2936]
 [3619 5944]]

Training Accuracy: 0.6594274432379073
Training TPR: 0.6215622712537906
Training FPR: 0.30318050392399837
```

In [123…
```
y_pred = lda.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Testing Confusion Matrix: \n", cm)
print ("\nTesting Accuracy:", accuracy_score(y_test, y_pred))
print ("Testing TPR:", recall_score(y_test, y_pred))
print ("Testing FPR:", FPR(y_test, y_pred))
```

```
Testing Confusion Matrix:
 [[2535 1691]
 [1927 2096]]

Testing Accuracy: 0.5614013819856952
Testing TPR: 0.5210042257022123
Testing FPR: 0.40014197823000475
```

Decision Tree Classifier with CV

In [135…
```
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.tree import DecisionTreeClassifier

grid_values = {'ccp_alpha': np.linspace(0, 0.0025, 10)}
grid_values

dtc = DecisionTreeClassifier(random_state=88)
dtc_cv = GridSearchCV(dtc, param_grid=grid_values, cv=10,verbose=3).fit(X_train, y_tra
```

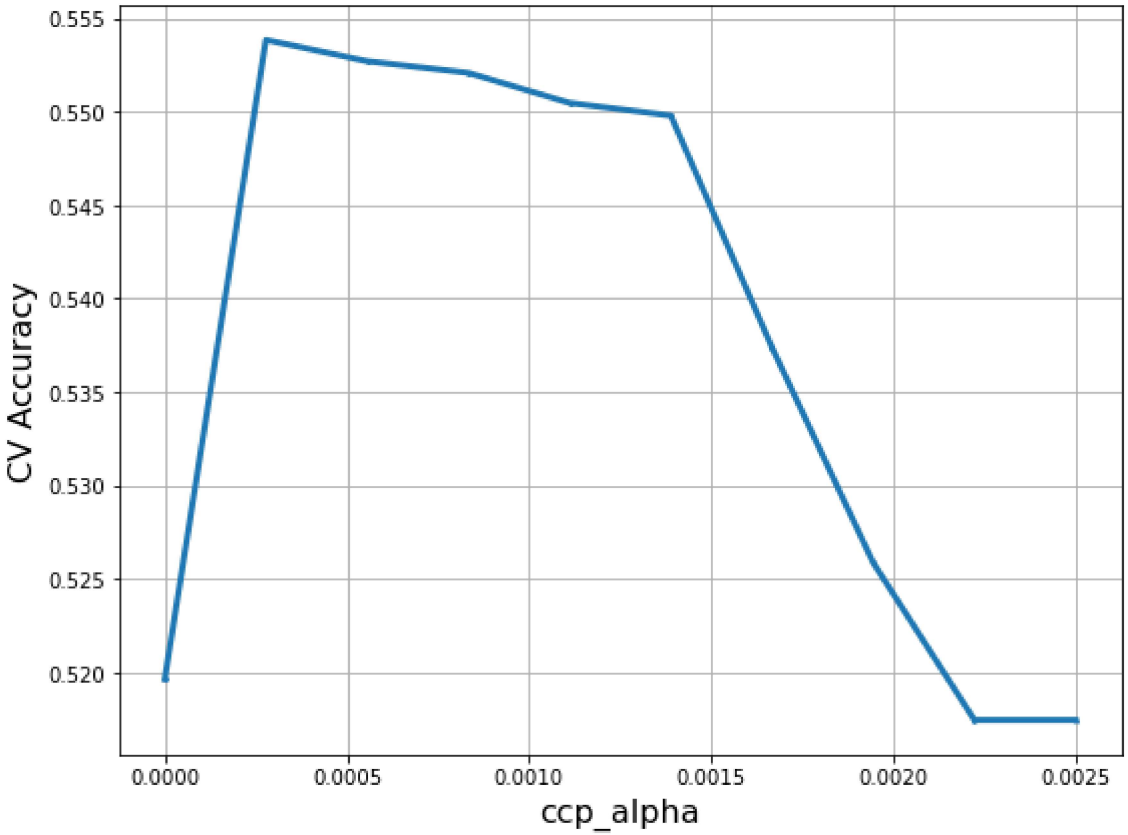
```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[CV 1/10] END .....................ccp_alpha=0.0;, score=0.507 total time=   9.8s
[CV 2/10] END .....................ccp_alpha=0.0;, score=0.514 total time=   9.9s
[CV 3/10] END .....................ccp_alpha=0.0;, score=0.528 total time=   8.5s
[CV 4/10] END .....................ccp_alpha=0.0;, score=0.509 total time=   7.5s
[CV 5/10] END .....................ccp_alpha=0.0;, score=0.524 total time=   8.3s
[CV 6/10] END .....................ccp_alpha=0.0;, score=0.527 total time=   8.8s
[CV 7/10] END .....................ccp_alpha=0.0;, score=0.516 total time=   8.5s
[CV 8/10] END .....................ccp_alpha=0.0;, score=0.517 total time=   9.0s
[CV 9/10] END .....................ccp_alpha=0.0;, score=0.527 total time=   8.0s
[CV 10/10] END ....................ccp_alpha=0.0;, score=0.529 total time=   8.9s
[CV 1/10] END ..ccp_alpha=0.0002777777777777778;, score=0.541 total time=   9.5s
[CV 2/10] END ..ccp_alpha=0.0002777777777777778;, score=0.558 total time=   8.1s
[CV 3/10] END ..ccp_alpha=0.0002777777777777778;, score=0.554 total time=   8.2s
[CV 4/10] END ..ccp_alpha=0.0002777777777777778;, score=0.568 total time=   7.7s
[CV 5/10] END ..ccp_alpha=0.0002777777777777778;, score=0.553 total time=   7.7s
[CV 6/10] END ..ccp_alpha=0.0002777777777777778;, score=0.559 total time=   8.2s
[CV 7/10] END ..ccp_alpha=0.0002777777777777778;, score=0.540 total time=   7.5s
[CV 8/10] END ..ccp_alpha=0.0002777777777777778;, score=0.558 total time=   9.5s
[CV 9/10] END ..ccp_alpha=0.0002777777777777778;, score=0.562 total time=   7.7s
[CV 10/10] END .ccp_alpha=0.0002777777777777778;, score=0.546 total time=   7.9s
[CV 1/10] END ..ccp_alpha=0.0005555555555555556;, score=0.552 total time=   7.8s
[CV 2/10] END ..ccp_alpha=0.0005555555555555556;, score=0.546 total time=   8.0s
[CV 3/10] END ..ccp_alpha=0.0005555555555555556;, score=0.562 total time=   8.1s
[CV 4/10] END ..ccp_alpha=0.0005555555555555556;, score=0.554 total time=   7.4s
[CV 5/10] END ..ccp_alpha=0.0005555555555555556;, score=0.563 total time=   7.5s
[CV 6/10] END ..ccp_alpha=0.0005555555555555556;, score=0.548 total time=   8.3s
[CV 7/10] END ..ccp_alpha=0.0005555555555555556;, score=0.551 total time=   7.5s
[CV 8/10] END ..ccp_alpha=0.0005555555555555556;, score=0.541 total time=   8.7s
[CV 9/10] END ..ccp_alpha=0.0005555555555555556;, score=0.551 total time=   9.5s
[CV 10/10] END .ccp_alpha=0.0005555555555555556;, score=0.560 total time=  11.1s
[CV 1/10] END ..ccp_alpha=0.0008333333333333333;, score=0.558 total time=  13.0s
[CV 2/10] END ..ccp_alpha=0.0008333333333333333;, score=0.552 total time=  11.1s
[CV 3/10] END ..ccp_alpha=0.0008333333333333333;, score=0.558 total time=   9.2s
[CV 4/10] END ..ccp_alpha=0.0008333333333333333;, score=0.555 total time=   8.5s
[CV 5/10] END ..ccp_alpha=0.0008333333333333333;, score=0.556 total time=   9.4s
[CV 6/10] END ..ccp_alpha=0.0008333333333333333;, score=0.544 total time=   8.8s
[CV 7/10] END ..ccp_alpha=0.0008333333333333333;, score=0.555 total time=   9.0s
[CV 8/10] END ..ccp_alpha=0.0008333333333333333;, score=0.540 total time=  10.3s
[CV 9/10] END ..ccp_alpha=0.0008333333333333333;, score=0.549 total time=   8.4s
[CV 10/10] END .ccp_alpha=0.0008333333333333333;, score=0.555 total time=   8.5s
[CV 1/10] END ..ccp_alpha=0.0011111111111111111;, score=0.561 total time=   8.4s
[CV 2/10] END ..ccp_alpha=0.0011111111111111111;, score=0.546 total time=   8.7s
[CV 3/10] END ..ccp_alpha=0.0011111111111111111;, score=0.554 total time=   8.4s
[CV 4/10] END ..ccp_alpha=0.0011111111111111111;, score=0.558 total time=   7.7s
[CV 5/10] END ..ccp_alpha=0.0011111111111111111;, score=0.545 total time=   8.8s
[CV 6/10] END ..ccp_alpha=0.0011111111111111111;, score=0.536 total time=  10.8s
[CV 7/10] END ..ccp_alpha=0.0011111111111111111;, score=0.555 total time=   8.0s
[CV 8/10] END ..ccp_alpha=0.0011111111111111111;, score=0.547 total time=  10.9s
[CV 9/10] END ..ccp_alpha=0.0011111111111111111;, score=0.548 total time=   9.2s
[CV 10/10] END .ccp_alpha=0.0011111111111111111;, score=0.555 total time=  10.8s
[CV 1/10] END ...ccp_alpha=0.0013888888888888889;, score=0.559 total time=  10.6s
[CV 2/10] END ...ccp_alpha=0.0013888888888888889;, score=0.546 total time=   9.5s
[CV 3/10] END ...ccp_alpha=0.0013888888888888889;, score=0.554 total time=  11.1s
[CV 4/10] END ...ccp_alpha=0.0013888888888888889;, score=0.555 total time=   7.9s
[CV 5/10] END ...ccp_alpha=0.0013888888888888889;, score=0.545 total time=   7.6s
[CV 6/10] END ...ccp_alpha=0.0013888888888888889;, score=0.536 total time=   8.2s
[CV 7/10] END ...ccp_alpha=0.0013888888888888889;, score=0.555 total time=   7.5s
[CV 8/10] END ...ccp_alpha=0.0013888888888888889;, score=0.547 total time=   8.6s
[CV 9/10] END ...ccp_alpha=0.0013888888888888889;, score=0.548 total time=   7.8s
```

```
[CV 10/10] END ..ccp_alpha=0.001388888888888889;, score=0.552 total time=   7.8s
[CV 1/10] END ..ccp_alpha=0.0016666666666666666;, score=0.542 total time=   7.9s
[CV 2/10] END ..ccp_alpha=0.0016666666666666666;, score=0.534 total time=   8.2s
[CV 3/10] END ..ccp_alpha=0.0016666666666666666;, score=0.536 total time=   8.3s
[CV 4/10] END ..ccp_alpha=0.0016666666666666666;, score=0.547 total time=   9.0s
[CV 5/10] END ..ccp_alpha=0.0016666666666666666;, score=0.534 total time=   8.0s
[CV 6/10] END ..ccp_alpha=0.0016666666666666666;, score=0.536 total time=   8.3s
[CV 7/10] END ..ccp_alpha=0.0016666666666666666;, score=0.530 total time=   7.5s
[CV 8/10] END ..ccp_alpha=0.0016666666666666666;, score=0.537 total time=   8.7s
[CV 9/10] END ..ccp_alpha=0.0016666666666666666;, score=0.538 total time=   7.9s
[CV 10/10] END .ccp_alpha=0.0016666666666666666;, score=0.539 total time=   7.9s
[CV 1/10] END ..ccp_alpha=0.0019444444444444444;, score=0.535 total time=   7.8s
[CV 2/10] END ..ccp_alpha=0.0019444444444444444;, score=0.515 total time=   8.1s
[CV 3/10] END ..ccp_alpha=0.0019444444444444444;, score=0.513 total time=   8.1s
[CV 4/10] END ..ccp_alpha=0.0019444444444444444;, score=0.521 total time=   7.5s
[CV 5/10] END ..ccp_alpha=0.0019444444444444444;, score=0.529 total time=   7.5s
[CV 6/10] END ..ccp_alpha=0.0019444444444444444;, score=0.526 total time=   8.2s
[CV 7/10] END ..ccp_alpha=0.0019444444444444444;, score=0.525 total time=   7.5s
[CV 8/10] END ..ccp_alpha=0.0019444444444444444;, score=0.531 total time=   8.6s
[CV 9/10] END ..ccp_alpha=0.0019444444444444444;, score=0.531 total time=   7.7s
[CV 10/10] END .ccp_alpha=0.0019444444444444444;, score=0.531 total time=   7.8s
[CV 1/10] END ..ccp_alpha=0.0022222222222222222;, score=0.524 total time=   7.8s
[CV 2/10] END ..ccp_alpha=0.0022222222222222222;, score=0.513 total time=   8.0s
[CV 3/10] END ..ccp_alpha=0.0022222222222222222;, score=0.513 total time=   9.0s
[CV 4/10] END ..ccp_alpha=0.0022222222222222222;, score=0.521 total time=   7.5s
[CV 5/10] END ..ccp_alpha=0.0022222222222222222;, score=0.517 total time=   7.5s
[CV 6/10] END ..ccp_alpha=0.0022222222222222222;, score=0.511 total time=   8.4s
[CV 7/10] END ..ccp_alpha=0.0022222222222222222;, score=0.516 total time=   7.5s
[CV 8/10] END ..ccp_alpha=0.0022222222222222222;, score=0.522 total time=   8.6s
[CV 9/10] END ..ccp_alpha=0.0022222222222222222;, score=0.519 total time=   7.7s
[CV 10/10] END .ccp_alpha=0.0022222222222222222;, score=0.518 total time=   7.8s
[CV 1/10] END .................ccp_alpha=0.0025;, score=0.524 total time=   7.8s
[CV 2/10] END .................ccp_alpha=0.0025;, score=0.513 total time=   8.0s
[CV 3/10] END .................ccp_alpha=0.0025;, score=0.513 total time=   8.0s
[CV 4/10] END .................ccp_alpha=0.0025;, score=0.521 total time=   7.4s
[CV 5/10] END .................ccp_alpha=0.0025;, score=0.517 total time=   7.5s
[CV 6/10] END .................ccp_alpha=0.0025;, score=0.511 total time=   8.2s
[CV 7/10] END .................ccp_alpha=0.0025;, score=0.516 total time=   7.5s
[CV 8/10] END .................ccp_alpha=0.0025;, score=0.522 total time=   8.6s
[CV 9/10] END .................ccp_alpha=0.0025;, score=0.519 total time=   7.7s
[CV 10/10] END ................ccp_alpha=0.0025;, score=0.518 total time=   8.0s
```

In [136…

```python
ccp_alpha = dtc_cv.cv_results_['param_ccp_alpha'].data
ACC_scores = dtc_cv.cv_results_['mean_test_score']
ccp_alpha
print(ACC_scores)

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_)

best_ccp = dtc_cv.best_params_["ccp_alpha"]
```

```
[0.51961414 0.55385277 0.55270918 0.55208537 0.55047528 0.54979982
 0.53738259 0.52584902 0.51743165 0.51743165]
```




```
Best ccp_alpha {'ccp_alpha': 0.0002777777777777778}
```

Runnning CART Model with best CCP Value

In [137…
```
best_ccp
```

Out[137]:
```
0.0002777777777777778
```

In [138…
```
#Train Regression Tree again based on best parameters
dtc_cv = DecisionTreeClassifier(ccp_alpha = best_ccp, random_state=88)
dtc_cv.fit(X_train, y_train)

y_pred = dtc_cv.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print ("Training Confusion Matrix: \n", cm)
print ("\nTraining Accuracy:", accuracy_score(y_train, y_pred))
print ("\nTraining TPR:", recall_score(y_train, y_pred))
print ("\nTraining FPR:", FPR(y_train, y_pred))
print()

y_pred = dtc_cv.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
print ("\nTesting TPR:", recall_score(y_test, y_pred))
print ("\nTesting FPR:", FPR(y_test, y_pred))
```

```
Training Confusion Matrix:
 [[7648 2036]
 [5719 3844]]

Training Accuracy: 0.5970800644256248

Training TPR: 0.4019659102792011

Training FPR: 0.21024370095002065

Confusion Matrix:
 [[3181 1045]
 [2619 1404]]

Accuracy: 0.5558249484786034

Testing TPR: 0.348993288590604

Testing FPR: 0.24727875059157595
```

Results of Learning

Logistic Regression:

Testing Accuracy: 56.29

Testing TPR: 52.82

Testing FPR: 40.39

LDA:

Testing Accuracy: 56.14

Testing TPR: 52.10

Testing FPR: 40.01

Best CART Model:

Accuracy: 55.58

Testing TPR: 34.89

Testing FPR: 24.72

Conclusion: As we can see from the above results for the three different models, we can see that the Testing Accuracy is higher for the Logistic Regression and LDA models than the CART models.

Since both are ~56% accuracy, we will look at the TPR as the secondary metric

We need to prioritize the TPR Metric over FPR for this problem statement since our aim is to maximize the number of useful questions up at the top. Therefore considering that the logistic regression model gives the best TPR, I consider it to be the best model among the three.

Ahead, I will perform the bootstrap procedure on the logistic regression model to check the variability of the results.

Boostrapping

Here is the function written for bootstrapping given test and train data, metrics required, sample size and random state.

In [108…
```
import time

def bootstrap_validation(test_data, test_label, train_label, model, metrics_list, samp
    tic = time.time()
    # Bootstrapping
    n_sample = sample
    n_metrics = len(metrics_list)
    output_array=np.zeros([n_sample, n_metrics])
    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), replace=Tru
        bs_data = test_data.loc[bs_index]
        bs_label = test_label.loc[bs_index]
        bs_predicted = model.predict(bs_data)
        for metrics_iter in range(n_metrics):
            metrics = metrics_list[metrics_iter]
            output_array[bs_iter, metrics_iter]=metrics(bs_predicted,bs_label)
        if bs_iter % 1000 == 0:
            print(bs_iter, time.time()-tic)

    output_df = pd.DataFrame(output_array)
    return output_df
```

Run Bootstrapping for Logistic regression model for Accuracy, TPR, and FPR scores

In [109…
```
bs_output_dtr = bootstrap_validation(X_test,y_test,y_train,logreg,
                                     metrics_list=[accuracy_score, recall_score,FPR],
                                     sample = 10000,random_state=88)
```

```
(10000, 3)
0 0.13199973106384277
1000 142.91167068481445
2000 287.6336669921875
3000 433.24766421318054
4000 579.3246660232544
5000 735.5926654338837
6000 931.779205083847
7000 1099.0136120319366
8000 1267.7159252166748
9000 1436.0110657215118
```

In [110…]
```
bs_output_dtr
```

Out[110]:

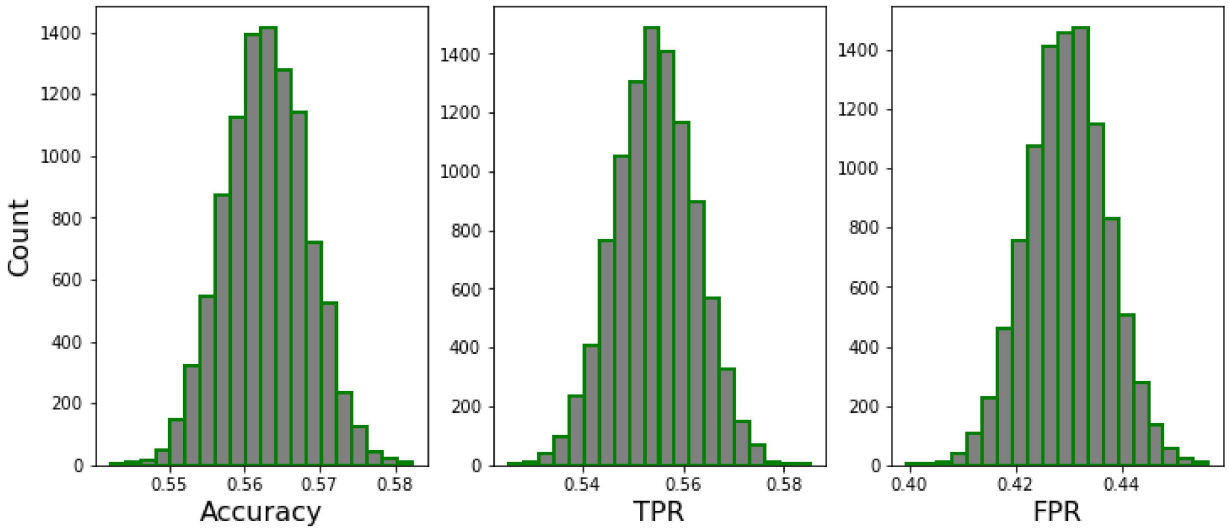
	0	1	2
0	0.563341	0.556985	0.431209
1	0.565887	0.560092	0.428932
2	0.557401	0.553128	0.439110
3	0.567341	0.573749	0.438344
4	0.560916	0.562467	0.440394
...	...	...	...
9995	0.564796	0.542061	0.414707
9996	0.565766	0.557566	0.427187
9997	0.561523	0.557884	0.435342
9998	0.566614	0.563069	0.430291
9999	0.558128	0.545146	0.430609

10000 rows × 3 columns

Plot Histogram of Metrics compiled over 10000 sample runs

In [111…]
```
fig, axs = plt.subplots(ncols=3, figsize=(12,5))
axs[0].set_xlabel('Accuracy', fontsize=16)
axs[1].set_xlabel('TPR', fontsize=16)
axs[2].set_xlabel('FPR', fontsize=16)

axs[0].set_ylabel('Count', fontsize=16)
axs[0].hist(bs_output_dtr.iloc[:,0], bins=20,edgecolor='green', linewidth=2,color = "g
axs[1].hist(bs_output_dtr.iloc[:,1], bins=20,edgecolor='green', linewidth=2,color = "g
axs[2].hist(bs_output_dtr.iloc[:,2], bins=20,edgecolor='green', linewidth=2,color = "g
```

Out[111]:
```
(array([   2.,    3.,   10.,   41.,  108.,  228.,  457.,  755., 1072.,
        1408., 1457., 1473., 1149.,  831.,  505.,  280.,  136.,   57.,
          19.,    9.]),
 array([0.3994484 , 0.40229695, 0.40514551, 0.40799406, 0.41084261,
        0.41369116, 0.41653972, 0.41938827, 0.42223682, 0.42508537,
        0.42793393, 0.43078248, 0.43363103, 0.43647958, 0.43932813,
        0.44217669, 0.44502524, 0.44787379, 0.45072234, 0.4535709 ,
        0.45641945]),
 <BarContainer object of 20 artists>)
```



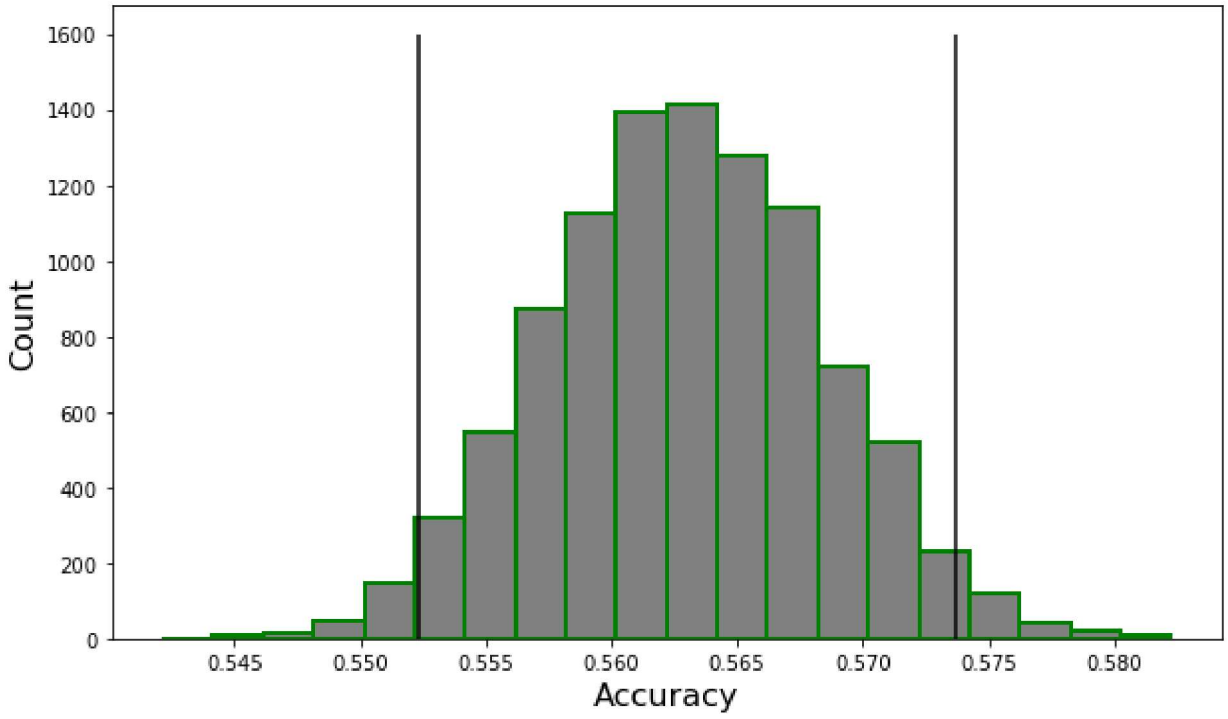
Plotting Confidence Intervals for Accuracy, TPR, FPR

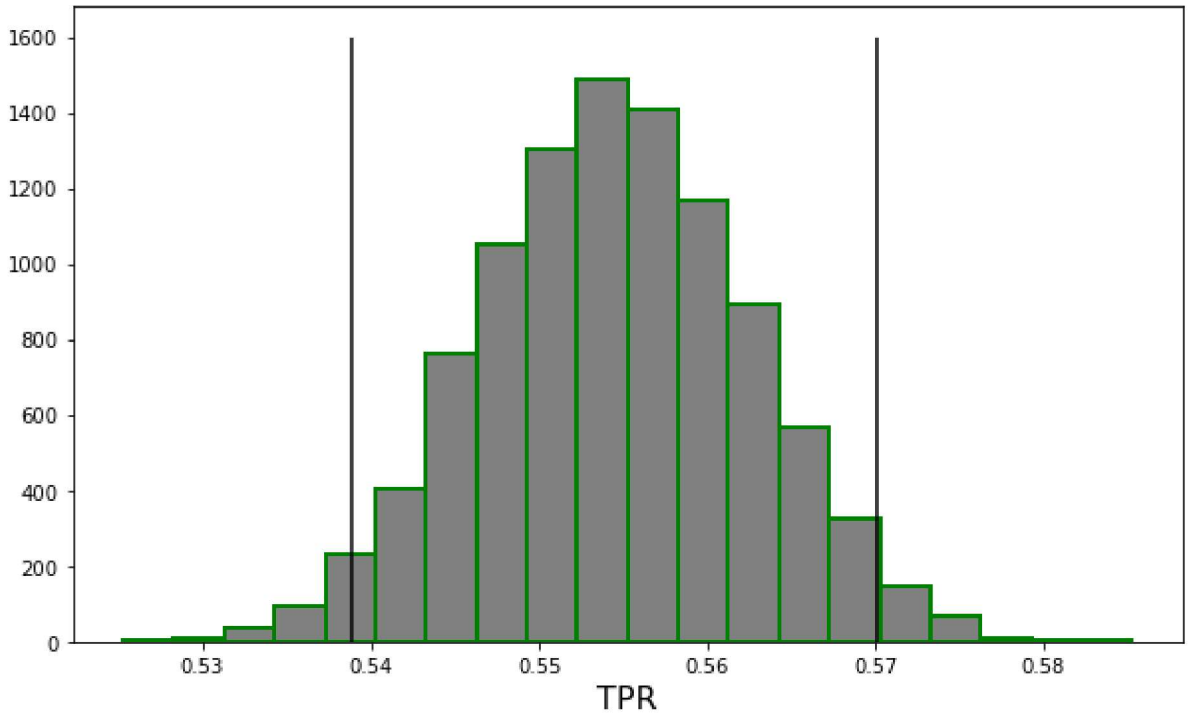
In [139…
```
CI1 = np.quantile(bs_output_dtr.iloc[:,0],np.array([0.025,0.975]))
print("The 95-percent confidence interval of Accuracy is %s" % CI1)

CI2 = np.quantile(bs_output_dtr.iloc[:,1],np.array([0.025,0.975]))
print("The 95-percent confidence interval of TPR is %s" % CI2)

CI3 = np.quantile(bs_output_dtr.iloc[:,2],np.array([0.025,0.975]))
print("The 95-percent confidence interval of FPR is %s" % CI3)
```

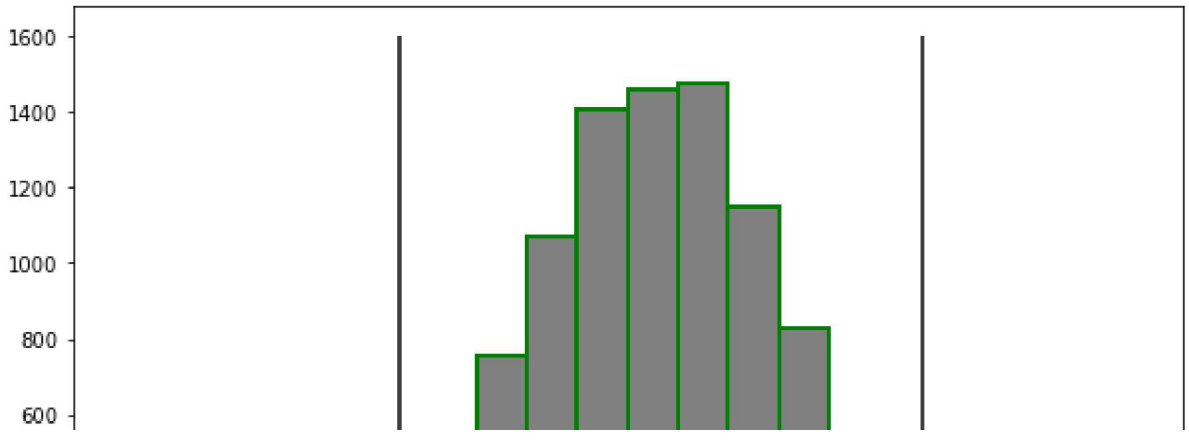
```
The 95-percent confidence interval of Accuracy is [0.55230937 0.57364529]
The 95-percent confidence interval of TPR is [0.53882338 0.5700872 ]
The 95-percent confidence interval of FPR is [0.41504656 0.44454447]
```

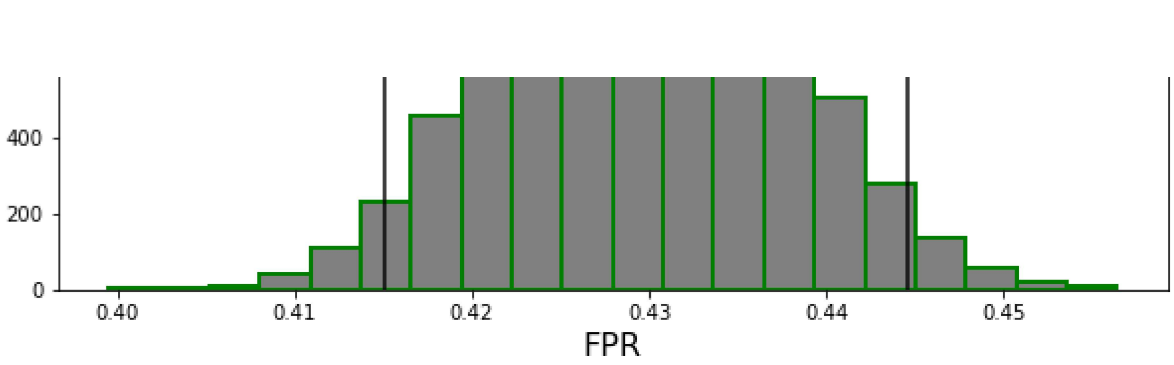
In [116…
```
fig, axs = plt.subplots(nrows=3, ncols=1, figsize=(10,20))
axs[0].set_ylabel('Count', fontsize=16)
axs[0].set_xlabel('Accuracy', fontsize=16)
axs[1].set_xlabel('TPR', fontsize=16)
axs[2].set_xlabel('FPR', fontsize=16)

axs[0].vlines(x=CI1[0], ymin = 0, ymax =1600, color = "black")
axs[0].vlines(x=CI1[1], ymin = 0, ymax =1600, color = "black")
axs[1].vlines(x=CI2[0], ymin = 0, ymax =1600, color = "black")
axs[1].vlines(x=CI2[1], ymin = 0, ymax =1600, color = "black")
axs[2].vlines(x=CI3[0], ymin = 0, ymax =1600, color = "black")
axs[2].vlines(x=CI3[1], ymin = 0, ymax =1600, color = "black")

axs[0].hist(bs_output_dtr.iloc[:,0], bins=20,edgecolor='green', linewidth=2,color = "g
axs[1].hist(bs_output_dtr.iloc[:,1], bins=20,edgecolor='green', linewidth=2,color = "g
axs[2].hist(bs_output_dtr.iloc[:,2], bins=20,edgecolor='green', linewidth=2,color = "g
```

Out[116]:
```
(array([   2.,    3.,   10.,   41.,  108.,  228.,  457.,  755., 1072.,
        1408., 1457., 1473., 1149.,  831.,  505.,  280.,  136.,   57.,
          19.,    9.]),
 array([0.3994484 , 0.40229695, 0.40514551, 0.40799406, 0.41084261,
        0.41369116, 0.41653972, 0.41938827, 0.42223682, 0.42508537,
        0.42793393, 0.43078248, 0.43363103, 0.43647958, 0.43932813,
        0.44217669, 0.44502524, 0.44787379, 0.45072234, 0.4535709 ,
        0.45641945]),
 <BarContainer object of 20 artists>)
```


Count
Accuracy
0
200
400
600
800
1000
1200
1400
1600
0.545
0.550
0.555
0.560
0.565
0.570
0.575
0.580

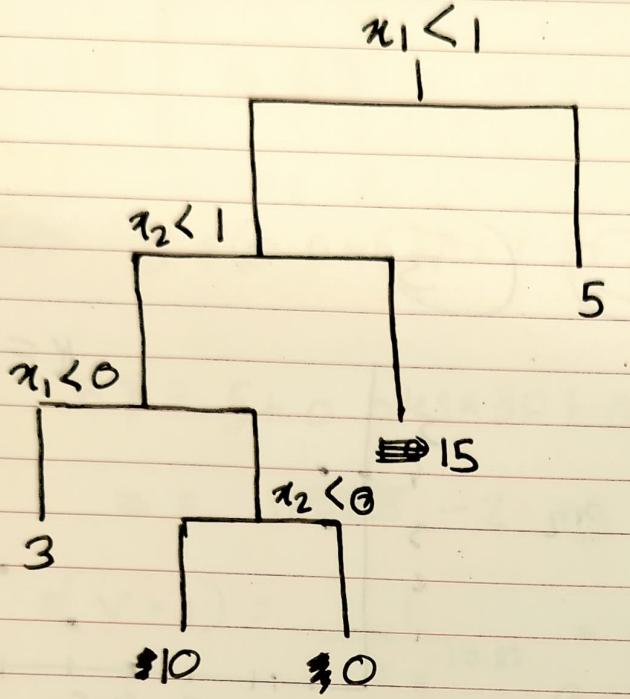

TPR
0
200
400
600
800
1000
1200
1400
1600
0.53
0.54
0.55
0.56
0.57
0.58


600
800
1000
1200
1400
1600

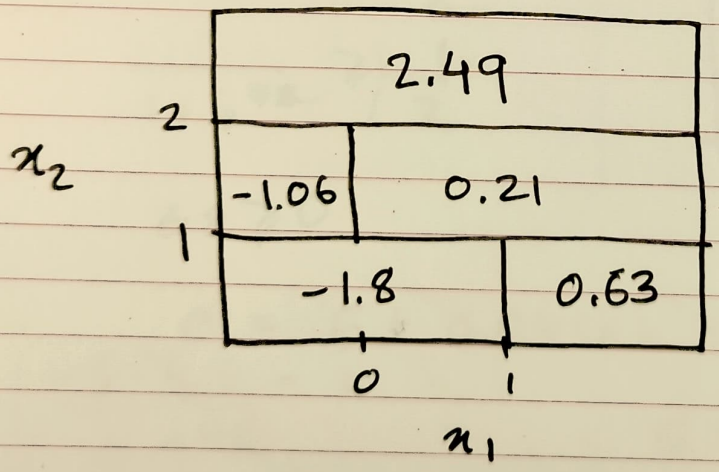

400
200
0
0.40
0.41
0.42
0.43
0.44
0.45
FPR

Problem 2)

a) Corresponding Tree:



b) Diagram:

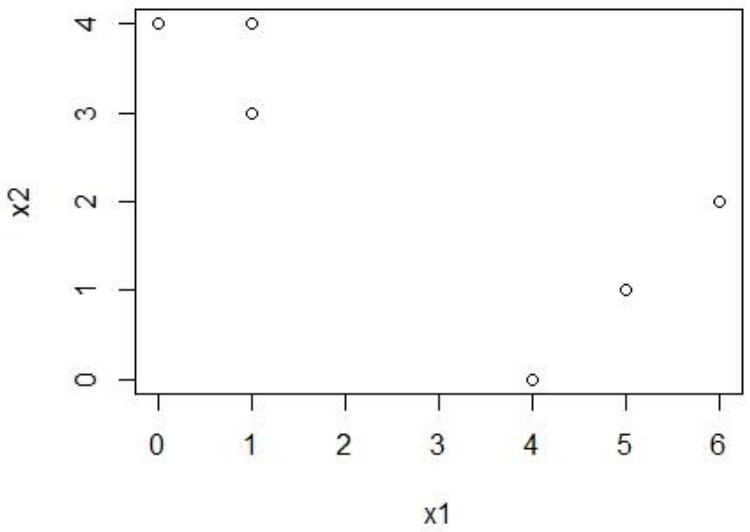


Problem 3)

```
> set.seed(1)
> x1 = c(1,1,0,5,6,4)
> x2 = c(4,3,4,1,2,0)
> x = cbind(x1, x2)
> x
     x1 x2
[1,]  1  4
[2,]  1  3
[3,]  0  4
[4,]  5  1
[5,]  6  2
[6,]  4  0
```

A)

```
> plot(x1, x2)
```

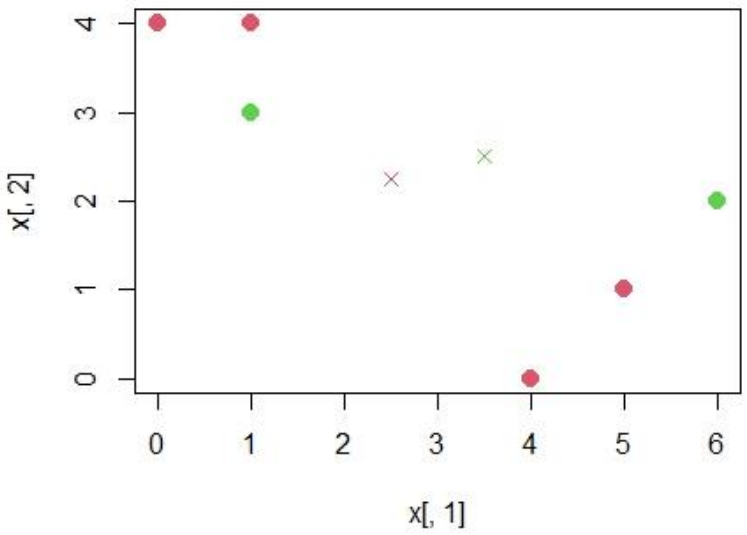


B)

```
> cluster = sample(2, nrow(x), replace=T)
> cluster
[1] 1 2 1 1 2 1
```

C)

```
> centroid1 = c(mean(x[cluster==1, 1]), mean(x[cluster==1, 2]))
> centroid2 = c(mean(x[cluster==2, 1]), mean(x[cluster==2, 2]))
> centroid1
[1] 2.50 2.25
> centroid2
[1] 3.5 2.5
> plot(x[, 1], x[, 2], col = (cluster + 1), pch = 20, cex = 2)
> points(centroid1[1], centroid1[2], col = 2, pch = 4)
> points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



D)

Function to calculate Euclidean distance:

```
> euclid = function(a, b) {
+   return(sqrt((a[1] - b[1])^2 + (a[2]-b[2])^2))
+ }
```

Function to calculate assign cluster:

```
> assign_cluster = function(x, centroid1, centroid2) {
+   cluster = rep(NA, nrow(x))
+   for (i in 1:nrow(x)) {
+     if (euclid(x[i,], centroid1) < euclid(x[i,], centroid2)) {
+       cluster[i] = 1
+     } else {
+       cluster[i] = 2
+     }
+   }
+   return(cluster)
+ }
```

Assign clusters:

```
> cluster = assign_cluster(x, centroid1, centroid2)
> cluster
[1] 1 1 1 2 2 2
```

E)

Function to repeat C and D:

```
> last_clusters = rep(-1, 6)
```

```
> while (!all(last_clusters == cluster)) {
+   last_clusters = cluster
+   centroid1 = c(mean(x[cluster==1, 1]), mean(x[cluster==1, 2]))
+   centroid2 = c(mean(x[cluster==2, 1]), mean(x[cluster==2, 2]))
+   print(centroid1)
+   print(centroid2)
+   cluster = assign_labels(x, centroid1, centroid2)
+ }
[1] 0.6666667 3.6666667
[1] 5 1

> cluster
[1] 1 1 1 2 2 2
```

F)

```
> plot(x[,1], x[,2], col=(cluster+1), pch=20, cex=2)
> points(centroid1[1], centroid1[2], col=1, pch=4)
> points(centroid2[1], centroid2[2], col=4, pch=4)
```