```
import sys
!{sys.executable} -m pip install -U pandas-profiling[notebook]
!jupyter nbextension enable --py widgetsnbextension
!pip install matplotlib
!pip install graphviz
```

```
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas!=1.4.0,<1.6,>1
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.9/dist-packages (from phik<0.13,>=0.11.1->
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.9/dist-packages (from pydantic<
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.1->jupy
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests<2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<2.29,>=
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.9/dist-packages (from requests<2.29,>=2
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests<2.29,>=2.24.0
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.9/dist-packages (from statsmodels<0.14,>=0.1
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.9/dist-packages (from widgetsnbextension~
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.9/dist-packages (from jedi>=0.10->ipy
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->
Requirement already satisfied: Send2Trash>=1.5.0 in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->widgets
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->widge
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->
Requirement already satisfied: nbformat in /usr/local/lib/python3.9/dist-packages (from notebook>=4.4.1->widgetsn
Requirement already satisfied: wcwidth in /usr/local/lib/python3.9/dist-packages (from prompt-toolkit<2.1.0,>=2.0
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.9/dist-packages (from pexpect->ipython>=
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.9/dist-packages (from argon2-cffi->n
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->noteboo
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->note
Requirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->w
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->not
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-packages (from nbconvert->note
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->noteb
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-packages (from nbformat->notebook>
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packages (from nbformat->notebook>=
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from argon2-cffi-bindings->a
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-packages (from beautifulsoup4->nbcon
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-packages (from bleach->nbconvert->not
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-packages (from cffi>=1.0.1->argon2-cffi-
Enabling notebook extension jupyter-js-widgets/extension...
Paths used for configuration of notebook:
        /root/.jupyter/nbconfig/notebook.json
Paths used for configuration of notebook:

      - Validating: OK
Paths used for configuration of notebook:
        /root/.jupyter/nbconfig/notebook.json
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (3.5.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (23.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (2
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (1.22.4)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (8.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (3.0.9
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (1.4
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (4.39
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matp
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packages (0.10.1)
```

```
import pandas as pd
import numpy as np

df= pd.read_csv("/content/drive/MyDrive/CIND 820 Capstone Project/merged_completedata.csv")
print(df.dtypes)
```

```
RecordID        int64
X             float64
Y             float64
```

```
FID             int64
BusinessID      int64
Name           object
Address        object
StreetNo        int64
StreetName     object
BldgNo         object
UnitNo         object
PostalCode     object
Location       object
Ward            int64
NAICSCode       int64
NAICSCat       object
NAICSDescr     object
Phone          object
Fax            object
TollFree       object
EMail          object
WebAddress     object
EmplRange       int64
CENT_X        float64
CENT_Y        float64
Year            int64
isnew          object
Closed         object
dtype: object
```

```
df = df[df['Year'] == 2019] #if we just look at closed from 2019
```

```
df.drop(['Year'], axis=1, inplace=True) #if we just look at closed from 2019
```

```
#NAICSCode back to object as it is nominal not ordinal
df['NAICSCode'] = df['NAICSCode'].astype(str)
```

```
# Let's display first 10 records
df.head(10)
```

	RecordID	X	Y	FID	BusinessID	Name	Address	StreetNo	StreetName	BldgNo	...	Phone	F
46689	46690	-79.665386	43.684736	1	7	Peel Car & Truck Rentals	7050 Bramalea Rd	7050	Bramalea Rd	Yes	...	905-670-2442	90 67 64
46690	46691	-79.642760	43.593515	2	4246	Real Fruit Bubble Tea	100 City Centre Dr	100	City Centre Dr	No	...		

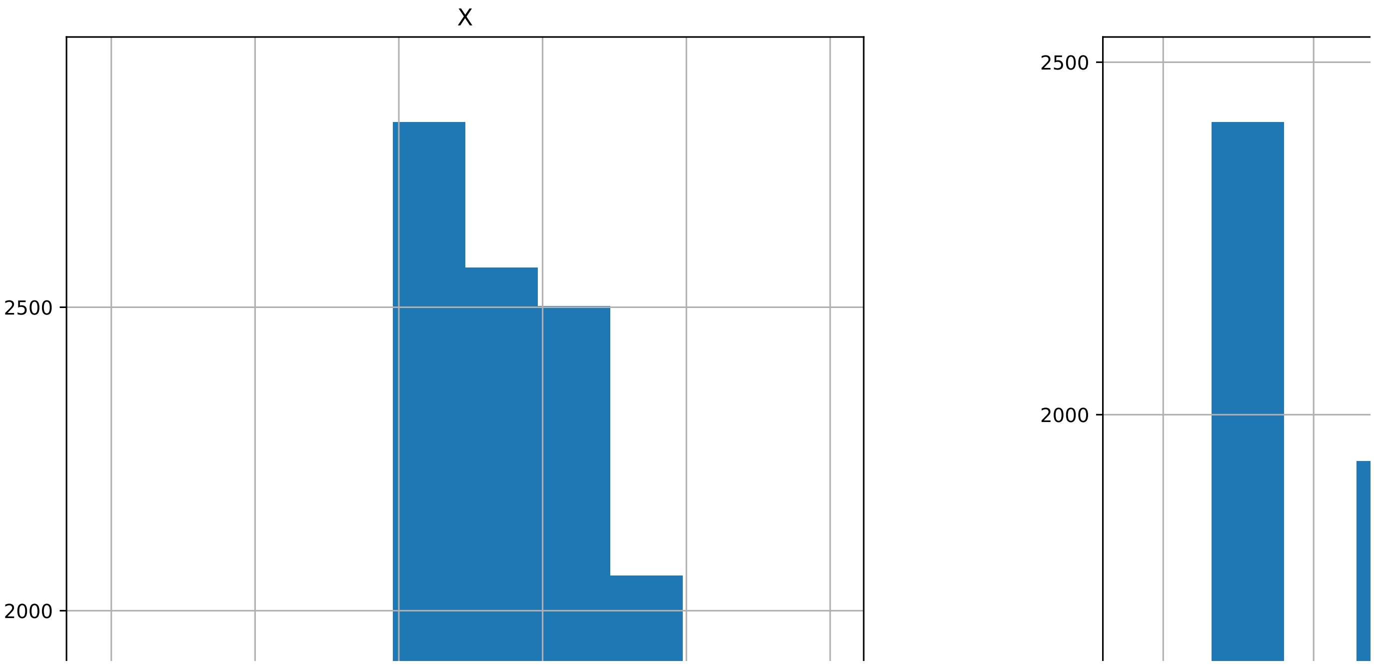
```
# look at meta information about data, such as null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16518 entries, 46689 to 63206
Data columns (total 27 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   RecordID    16518 non-null  int64  
 1   X           16518 non-null  float64
 2   Y           16518 non-null  float64
 3   FID         16518 non-null  int64  
 4   BusinessID  16518 non-null  int64  
 5   Name        16518 non-null  object 
 6   Address     16518 non-null  object 
 7   StreetNo    16518 non-null  int64  
 8   StreetName  16518 non-null  object 
 9   BldgNo      16518 non-null  object 
 10  UnitNo      16518 non-null  object 
 11  PostalCode  16518 non-null  object 
 12  Location    16518 non-null  object 
 13  Ward        16518 non-null  int64  
 14  NAICSCode   16518 non-null  object 
 15  NAICSCat    16518 non-null  object 
 16  NAICSDescr  16518 non-null  object 
 17  Phone       16518 non-null  object 
 18  Fax         16518 non-null  object 
 19  TollFree    16518 non-null  object 
 20  EMail       16518 non-null  object 
 21  WebAddress  16518 non-null  object 
 22  EmplRange   16518 non-null  int64  
 23  CENT_X      16518 non-null  float64
 24  CENT_Y      16518 non-null  float64
 25  isnew       16518 non-null  object 
 26  Closed      16518 non-null  object 
dtypes: float64(4), int64(6), object(17)
memory usage: 3.5+ MB
```

```
# Let's see meta information about numeric data, we can also see if there any extreme values
df.describe()
```

	RecordID	X	Y	FID	BusinessID	StreetNo	Ward	EmplRange	CE
count	16518.000000	16518.000000	16518.000000	16518.000000	16518.000000	16518.000000	16518.000000	16518.000000	16518.00
mean	54948.500000	-79.657689	43.601356	8259.500000	38317.374803	2949.933285	5.372927	2.183981	608803.89
std	4768.480209	0.046612	0.056180	4768.480209	32183.768108	2364.551005	2.452044	1.450311	3622.95
min	46690.000000	-79.802980	43.485170	1.000000	7.000000	1.000000	1.000000	1.000000	596627.93
25%	50819.250000	-79.697599	43.560065	4130.250000	10300.250000	1050.250000	5.000000	1.000000	606962.72
50%	54948.500000	-79.655443	43.600388	8259.500000	20467.000000	2380.000000	5.000000	2.000000	609549.25
75%	59077.750000	-79.622510	43.643674	12388.750000	57398.250000	5144.500000	7.000000	3.000000	611124.36
max	63207.000000	-79.550935	43.732864	16518.000000	93823.000000	7895.000000	11.000000	9.000000	616985.05

```
df.drop(['RecordID','FID','BusinessID','StreetNo','Ward','CENT_X','CENT_Y','EmplRange'], axis=1, inplace=True)



# First split the data into train and test set
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
# Our class column is Closed and everything else will be used as features
class_col_name='Closed'

feature_names=df.columns[df.columns != class_col_name ]
# 80% training and 20% test
X_train, X_test, y_train, y_test = train_test_split(df.loc[:, feature_names], df[class_col_name], test_size=0.2,random_


#plot the distribution of continuous variables

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

X_train[X_train.dtypes[(X_train.dtypes=="float64")|(X_train.dtypes=="int64")]
                        .index.values].hist(figsize=[17,17])
from IPython.display import set_matplotlib_formats # for better resolution in pdf export
set_matplotlib_formats('pdf', 'svg')
```




```
#change categorical values to numeric using LabelEncoder
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
# Iterating over all the common columns in train and test
for col in X_test.columns.values:
    # Encoding only categorical variables
    if X_test[col].dtypes=='object':
    # Using whole data to form an exhaustive list of levels
      data=X_train[col].append(X_test[col])
      le.fit(data.values)
      X_train[col]=le.transform(X_train[col])
      X_test[col]=le.transform(X_test[col])

#4 minutes 24 seconds to run this
```

```
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
<ipython-input-275-9b0bb92014d6>:9: FutureWarning: The series.append method is deprecated and will be removed fro
  data=X_train[col].append(X_test[col])
```

```
#help(tree.DecisionTreeClassifier)


#Decision tree 5 levels deep
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth=5)
clf = clf.fit(X_train, y_train)
print("Successfuly trained the decision tree...")
```

```
Successfuly trained the decision tree...
```

```
#plot the decision tree
import graphviz
#Get unique class values to display on the tree
class_values=df[class_col_name].unique()
print ("Class Names",class_values)


dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=feature_names,
                                class_names=class_values,
                                filled=True)
# Plot tree
graph = graphviz.Source(dot_data, format="png")
graph
```

```
Class Names ['No' 'Yes']
```

```
# Let's make the prdictions on the test set that we set aside earlier using the trained tree
y_pred = clf.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cf=confusion_matrix(y_test, y_pred)
print ("Confusion Matrix")
print(cf)
tn, fp, fn, tp=cf.ravel()
print ("TP: ", tp,", FP: ", fp,", TN: ", tn,", FN:", fn)
```

```
Confusion Matrix
[[2752   10]
 [ 538    4]]
TP:  4 , FP:  10 , TN:  2752 , FN: 538
```


```
#print precision, recall, and accuracy from the perspective of each of the class (0 and 1 for the dataset)
from sklearn.metrics import classification_report
from sklearn import metrics

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

              No       0.84      1.00      0.91      2762
             Yes       0.29      0.01      0.01       542

        accuracy                           0.83      3304
       macro avg       0.56      0.50      0.46      3304
    weighted avg       0.75      0.83      0.76      3304
```


```
#Similarly we can Train and Test in Bayes
#from sklearn.naive_bayes import MultinomialNB
#Since we have negative values in the X column use GaussianNB instead of MultinomialNB
from sklearn.naive_bayes import GaussianNB
######from sklearn.naive_bayes import MultinomialNB

#Create a MultiNomial NB Classifier
#####nb = MultinomialNB()
nb = GaussianNB()

#Train the model using the training sets
nb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = nb.predict(X_test)
```

```
print ("Total Columns (including class)",len(df.columns))
```

```
Total Columns (including class) 19
```

```
#print("Number of features used ",nb.n_features_)
print("Classes ",nb.classes_)
print("Number of records for classes ",nb.class_count_)
#print("Log prior probability for classes ", nb.class_log_prior_)
print("Probability of each class", nb.class_prior_)
#print("Log conditional probability for each feature given a class\n",nb.feature_log_prob_)
print("float absolutive additive value to variances", nb.epsilon_)
#print("array,shape variance of each feature per class",nb.sigma_)
print("array,shape mean of each feature per class",nb.theta_)
```

```
Classes  ['No' 'Yes']
Number of records for classes  [11042.  2172.]
Probability of each class [0.83562888 0.16437112]
float absolutive additive value to variances 0.021032330650500256
array,shape mean of each feature per class [[-7.96575960e+01  4.36025371e+01  7.72517171e+03  2.89902373e+03
   2.74015577e+02  5.11682666e-02  6.55859446e-01  1.32051259e+01
   3.16664554e+01  1.31677232e+01  9.71418221e+00  3.22962145e+02
   7.66111212e+03  2.82494584e+03  1.49972831e-01  6.10215541e-01
   7.42800217e-01  1.00978084e-01]
 [-7.96598839e+01  4.35966464e+01  7.57958057e+03  2.84230617e+03
   2.75028085e+02  4.41988950e-02  7.05340700e-01  1.33462247e+01
   3.40626151e+01  1.23678637e+01  1.02808471e+01  3.22968232e+02
   7.21436326e+03  2.51757689e+03  1.66666667e-01  5.92081031e-01
   7.23296501e-01  1.36740331e-01]]
```

```
from sklearn.metrics import classification_report
from sklearn import metrics

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

              No       0.84      1.00      0.91      2762
             Yes       0.19      0.01      0.01       542

        accuracy                           0.83      3304
       macro avg       0.51      0.50      0.46      3304
    weighted avg       0.73      0.83      0.76      3304
```

```
#decribe categorical data
df.describe(include='O')
```

	Name	Address	StreetName	BldgNo	UnitNo	PostalCode	Location	NAICSCode	NAICSCat	NAICSDescr	Phone	F
count	16518	16518	16518	16518	16518	16518	16518	16518	16518	16518	16518	16
unique	15233	5959	598	2	2	34	56	24	19	687	15531	9
top	Subway	100 City Centre Dr	Dundas St E	No	Yes	L4W	Northeast EA (West)	81	Retail Trade	Limited-service eating places		
freq	43	222	667	15689	10986	2714	4131	1873	2303	808	403	6

```
#drop columns that have unique values
#df.drop(['Name','Address','StreetName','NAICSDescr'], axis=1, inplace=True)
df.drop(['Name','Address','StreetName','NAICSDescr',"PostalCode", "BldgNo", "NAICSCat","NAICSCode", "Phone", "Fax","Tol
```

```
#Let's create a list for our categorical columns for the dataset, we need this later
#cat_cols=["PostalCode", "BldgNo", "UnitNo","NAICSCat","NAICSCode", "Phone", "Fax","TollFree","WebAddress","isnew","EMa
cat_cols=["UnitNo","WebAddress","isnew","EMail","Location"]
```

```
# Create a copy of the data frame in memory with a different name
df_onehot=df.copy()
#convert only categorical variables/features to dummy/one-hot features
df_onehot = pd.get_dummies(df, columns=cat_cols, prefix = cat_cols)
#print the dataset
df_onehot
```

	X	Y	Closed	UnitNo_No	UnitNo_Yes	WebAddress_No	WebAddress_Yes	isnew_No	isnew_Yes	EMail_
46689	-79.665386	43.684736	No	0	1	0	1	1	0	
46690	-79.642760	43.593515	No	0	1	0	1	0	1	
46691	-79.667311	43.682752	No	0	1	0	1	1	0	
46692	-79.629235	43.698932	No	0	1	0	1	0	1	
46693	-79.629235	43.698932	No	0	1	1	0	0	1	
...	...	...	...	...	...	...	...	...	...	...
63202	-79.697599	43.517559	Yes	1	0	0	1	0	1	
63203	-79.697599	43.517559	No	1	0	0	1	0	1	
63204	-79.697599	43.517559	Yes	1	0	0	1	0	1	
63205	-79.697599	43.517559	No	0	1	1	0	0	1	

```
#X column has negative values. Those column values won't work with Naive Bayes. So run the following
# line of code for the data set to make negative values to 0. Note that it is not necessary for decision tree to remove
df_onehot["X"]=df_onehot["X"].apply(lambda x: 0 if x<0 else x)
```

```
#Repeat the train test set split
from sklearn.model_selection import train_test_split

# Uncomment following line for class name for the dataset
class_col_name="Closed"

one_hot_feature_names=df_onehot.columns[df_onehot.columns != class_col_name]
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(df_onehot.loc[:, one_hot_feature_names], df_onehot[class_col_name],
```

```
# Repeat Naive Bayes modeling
from sklearn.naive_bayes import MultinomialNB

#Create a MultiNomial NB Classifier
nb = MultinomialNB()
print ("Succesfully NB done..")
#Train the model using the training sets
nb.fit(X_train, y_train)
print ("Succesfully fit done..")
#Predict the response for test dataset
y_pred = nb.predict(X_test)
print ("Succesfully done..")
```

```
Succesfully NB done..
Succesfully fit done..
Succesfully done..
```

```
#print("Number of features used ",nb.n_features_)
print("Classes ",nb.classes_)
print("Number of records for classes ",nb.class_count_)
print("Log prior probability for classes ", nb.class_log_prior_)
print("Log conditional probability for each feature given a class\n",nb.feature_log_prob_)
```

```
Classes  ['No' 'Yes']
Number of records for classes  [11036.  2178.]
Log prior probability for classes  [-0.18011422 -1.80286985]
Log conditional probability for each feature given a class
 [[-13.19270961  -0.10868217  -4.95232449  -4.30436674  -5.22959755
   -4.1850976   -3.99282369  -6.15217322  -4.833809    -4.37245324
   -7.63202798  -8.51988077  -8.01092606  -8.29486981  -8.94421437
  -13.19270961  -9.00305487  -8.48317941  -9.93461307  -9.03382652
   -8.94421437  -7.79908206  -9.55512345  -7.65537534  -7.14062044
  -10.05721539  -8.71537279  -6.70960226  -7.90950588  -9.03382652
   -9.01832234  -8.9732019   -8.23688255  -6.77761265  -8.34852252
   -8.6818501  -10.19697733  -8.02222561  -9.20372556  -9.06557522
```

```
        -8.64941483  -7.7288778   -6.89376036  -9.3215086   -9.14965834
        -9.38604712  -9.55512345  -9.97383378  -7.6281892   -5.26466401
        -8.24394972  -9.16735792  -9.7269737   -9.14965834  -8.82326176
        -9.18537642  -9.30088931  -9.97383378  -9.50383015  -8.34067934
       -11.5832717   -8.11130524  -9.26088397 -11.80641525  -8.88864451
        -5.84863676]
      [-11.57034002  -0.10918715  -5.09029546  -4.23862505  -5.18514562
        -4.2007393   -4.02865692  -5.88336467  -4.76240508  -4.41963857
        -7.23960668  -8.86228982  -8.62590104  -8.73712668  -9.49089848
       -11.57034002  -9.96090211  -8.93128269  -9.26775493  -8.93128269
        -9.00539067  -7.74169863  -9.26775493  -8.01499196  -7.12768877
       -10.47172773  -8.67996827  -6.85184115  -8.3514642   -9.49089848
        -9.37311545  -9.37311545  -8.20304419  -6.82540789  -8.3514642
        -9.00539067  -9.62442987  -8.0439795  -11.57034002  -9.17244475
        -9.26775493  -7.72019242  -6.67999089  -9.00539067  -9.26775493
       -10.18404566  -9.08543337 -10.87719284  -7.95942211  -5.25317534
        -8.31224349  -9.49089848  -9.37311545  -9.00539067  -9.49089848
        -8.7977513   -9.49089848 -10.18404566 -10.47172773  -7.95942211
       -11.57034002  -8.20304419  -9.77858055 -11.57034002  -8.73712668
        -5.48584061]]
```

```
from sklearn.metrics import confusion_matrix
cf=confusion_matrix(y_test, y_pred)
print ("Confusion Matrix")
print(cf)
tn, fp, fn, tp=cf.ravel()
print ("TP: ", tp,", FP: ", fp,", TN: ", tn,", FN:", fn)
```

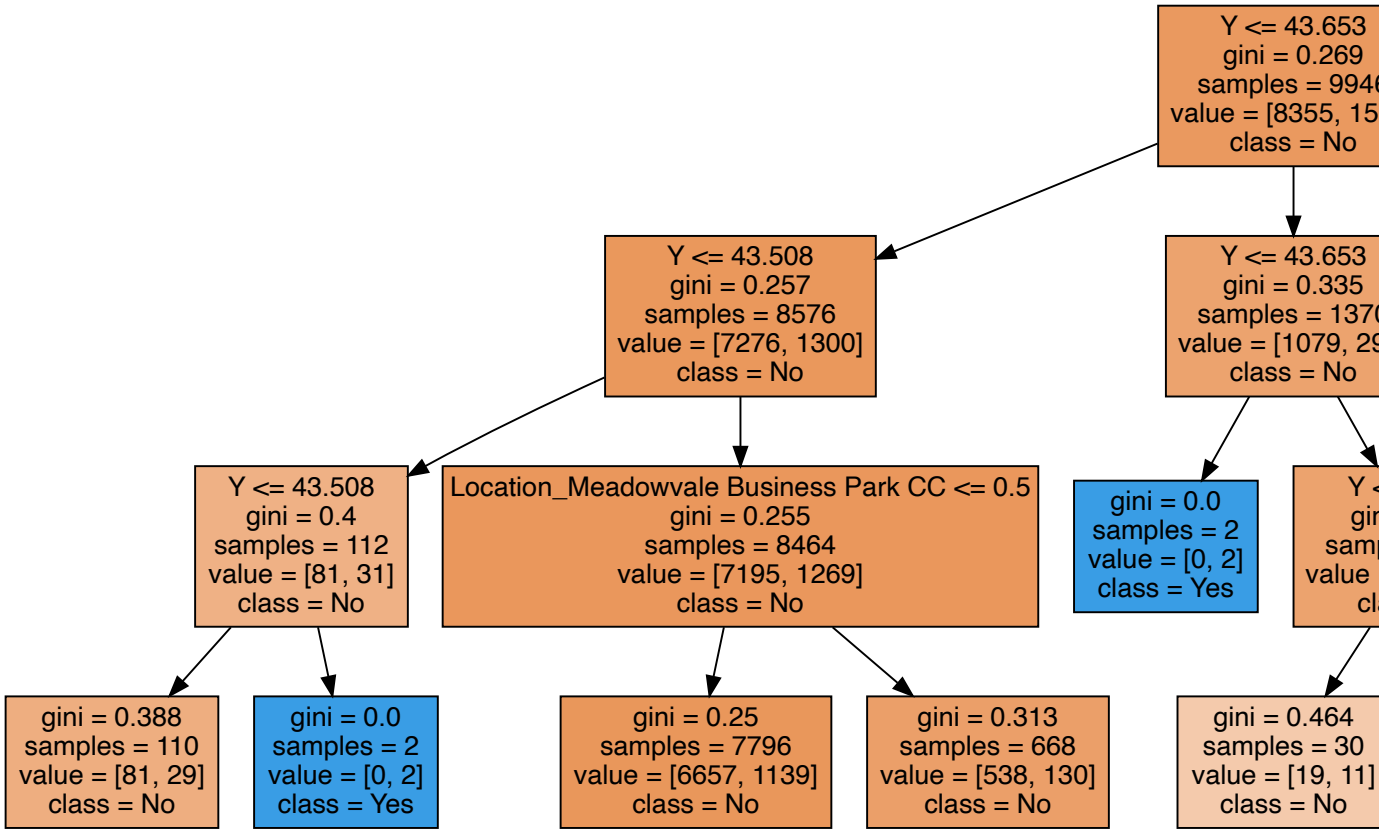
```
Confusion Matrix
[[2768    0]
 [ 536    0]]
TP:  0 , FP:  0 , TN:  2768 , FN: 536
```

```
from sklearn.metrics import classification_report
from sklearn import metrics

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

              No       0.84      1.00      0.91      2768
             Yes       0.00      0.00      0.00       536

        accuracy                           0.84      3304
       macro avg       0.42      0.50      0.46      3304
    weighted avg       0.70      0.84      0.76      3304

    /usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision
      _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth=5)
clf = clf.fit(X_train, y_train)
import graphviz
#Get unique class values to display on the tree
class_values=df_onehot[class_col_name].unique()
print ("class Names",class_values)

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=one_hot_feature_names,
                                class_names=class_values,
                                filled=True)

# Draw graph
graph = graphviz.Source(dot_data, format="png")
graph
```

```
class Names ['No' 'Yes']
```




```
# Perform prediction on the test set
y_pred = clf.predict(X_test)


# Get classification report
from sklearn.metrics import classification_report
from sklearn import metrics

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

              No       0.84      1.00      0.91      2768
             Yes       0.29      0.00      0.01       536

        accuracy                           0.84      3304
       macro avg       0.56      0.50      0.46      3304
    weighted avg       0.75      0.84      0.76      3304
```

✓ 0s completed at 8:00 PM