IMPORTING PYTHON PACKAGES

```
#Import Python Packages
#from google.colab import drive
#drive.mount('/content/drive/')

from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Mounted at /gdrive
/gdrive
```

IMPORT ALL NECESSARY LIBRARIES

```
#Import all necessary librabry

import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn import tree
```

READ TRAINING DATA FILE

```
trainfile = r'/gdrive/My Drive/Santander Customer Satisfaction - TRAIN.csv'
trainData = pd.read_csv(trainfile)
trainData.head()
```

READ TEST DATA FILE

```
testfile = r'/gdrive/My Drive/Santander Customer Satisfaction - TEST-Without TARGET.csv'
testData = pd.read_csv(testfile)
testData.head()
```

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult
0	2	2	32	0.0	0.0	0.
1	5	2	35	0.0	0.0	0.
2	6	2	23	0.0	0.0	0.
3	7	2	24	0.0	0.0	0.
4	9	2	23	0.0	0.0	0.

5 rows × 370 columns

DATA SHAPE

```
print(trainData.shape)      # To get (Number of Rows, Number of Columns) of a data frame we u
print(testData.shape)
```

```
(76020, 371)
(75818, 370)
```

COLUMN INFORMATION

```
#Understanding the Columns

trainData.info()
print()
testData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76020 entries, 0 to 76019
Columns: 371 entries, ID to TARGET
dtypes: float64(111), int64(260)
memory usage: 215.2 MB

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75818 entries, 0 to 75817
Columns: 370 entries, ID to var38
```

```
dtypes: float64(110), int64(260)
memory usage: 214.0 MB
```

CHECKING FOR MISSING VALUES

```
# To check number of null values
trainData.isna().sum().sort_values(ascending=False)
```

```
ID                         0
imp_trasp_var17_in_ult1    0
ind_var7_emit_ult1         0
imp_venta_var44_ult1       0
imp_venta_var44_hace3      0
                          ..
num_op_var40_hace3         0
num_op_var40_hace2         0
num_var25                  0
num_var25_0                0
TARGET                     0
Length: 371, dtype: int64
```

DESCRIBE DATA

```
# To check basic statistics of a data set, column wise
trainData.describe()
```

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_come
count	76020.000000	76020.000000	76020.000000	76020.000000	76020
mean	75964.050723	-1523.199277	33.212865	86.208265	72
std	43781.947379	39033.462364	12.956486	1614.757313	339
min	1.000000	-999999.000000	5.000000	0.000000	0
25%	38104.750000	2.000000	23.000000	0.000000	0
50%	76043.000000	2.000000	28.000000	0.000000	0
75%	113748.750000	2.000000	40.000000	0.000000	0
max	151838.000000	238.000000	105.000000	210000.000000	12888

8 rows × 371 columns

COLUMN NAMES

```
#To get list of names of all Columns from a dataframe

TrainCols = list(trainData.columns.values)
TestCols = list(testData.columns.values)
print(TrainCols)
print(TestCols)
```

```
['ID', 'var3', 'var15', 'imp_ent_var16_ult1', 'imp_op_var39_comer_ult1', 'imp_op_var39_c
['ID', 'var3', 'var15', 'imp_ent_var16_ult1', 'imp_op_var39_comer_ult1', 'imp_op_var39_c
```

SEPARATE TARGET COLUMN FROM TRAINING DATA SET

```
# Seperate Target column from Train Data
Xtrain = trainData[TrainCols[0:len(TrainCols)-1]].copy()
Ytrain = trainData[['TARGET']].copy()
print(Xtrain.shape)
print(Ytrain.shape)
Xtest = testData.copy()
```

```
(76020, 370)
(76020, 1)
```

INTIALISE DECISION TREES

```
dt = DecisionTreeClassifier()
dt.fit(Xtrain, Ytrain)
```

```
DecisionTreeClassifier()
```

BASIC ANALYSIS

```
print("Count of 0 & 1 in TARGET column for Train Data")
print(Ytrain['TARGET'].value_counts())
```

```
Count of 0 & 1 in TARGET column for Train Data
0    73012
1     3008
Name: TARGET, dtype: int64
```

CALCULATE ACCURACY FOR TRAINING DATA

```
X_Pred = dt.predict(Xtrain)
#Model Accuracy
```

```
print("Accuracy:", metrics.accuracy_score(Ytrain,X_Pred))
```

```
Accuracy: 1.0
```

DATA SPLITTING

```
X_train, X_test, Y_train, Y_test = train_test_split(Xtrain, Ytrain, test_size = .3)
```

MODEL 1

```
dt1 = DecisionTreeClassifier(criterion='gini',splitter='random',class_weight='balanced',max_d
)
dt1 = dt1.fit(X_train,Y_train)
Y_Pred = dt1.predict(Xtest)
Y_Pred = pd.DataFrame(Y_Pred,columns=['TARGET'])
Y_Pred_test = dt1.predict(X_test)
print("Accuracy:", metrics.accuracy_score(Y_test,Y_Pred_test))
```

```
Accuracy: 0.7575199508901166
```

MODEL 2

```
dt2 = DecisionTreeClassifier(criterion='entropy',splitter='random',class_weight='balanced',ma
)
dt2 = dt2.fit(X_train,Y_train)
Y_Pred = dt2.predict(Xtest)
Y_Pred = pd.DataFrame(Y_Pred,columns=['TARGET'])
Y_Pred_test = dt2.predict(X_test)
print("Accuracy:", metrics.accuracy_score(Y_test,Y_Pred_test))
```

```
Accuracy: 0.7060422695781812
```

MODEL 3

```
dt3 = DecisionTreeClassifier(criterion='gini',splitter='best',class_weight='balanced',max_dep
)
dt3 = dt3.fit(X_train,Y_train)
Y_Pred = dt3.predict(Xtest)
Y_Pred = pd.DataFrame(Y_Pred,columns=['TARGET'])
Y_Pred_test = dt3.predict(X_test)
print("Accuracy:", metrics.accuracy_score(Y_test,Y_Pred_test))
```

```
Accuracy: 0.7684381303165834
```

MODEL 4

```
dt4 = DecisionTreeClassifier(criterion='entropy',splitter='best',class_weight='balanced',max_
)
dt4 = dt4.fit(X_train,Y_train)
Y_Pred = dt4.predict(Xtest)
Y_Pred = pd.DataFrame(Y_Pred,columns=['TARGET'])
Y_Pred_test = dt4.predict(X_test)
print("Accuracy:", metrics.accuracy_score(Y_test,Y_Pred_test))
```

```
Accuracy: 0.7863720073664825
```

KAGGLE PREDICTION

```
PreID=Xtest['ID']
Y_Pred_prob_test=dt4.predict_proba(Xtest)
Y_Pred_prob_test = pd.DataFrame(Y_Pred_prob_test[:,1],columns=['TARGET'])
pd.concat([PreID,Y_Pred_prob_test],axis=1).to_csv("/gdrive/My Drive/ResultCIS508-4.csv",index
```

PLOTTING DECISION TREE

```
tree.plot_tree(dt)
```

```
[Text(0.5219153298451271, 0.9915254237288136, 'X[183] <= 2.955\ngini =
0.076\nsamples = 76020\nvalue = [73012, 3008]'),
 Text(0.2623390231753928, 0.9745762711864406, 'X[2] <= 27.5\ngini = 0.161\nsamples =
21289\nvalue = [19403, 1886]'),
 Text(0.06640142649461662, 0.9576271186440678, 'X[369] <= 56607.525\ngini =
0.048\nsamples = 11140\nvalue = [10868, 272]'),
 Text(0.024471066131690113, 0.940677966101695, 'X[2] <= 25.5\ngini = 0.112\nsamples
= 1441\nvalue = [1355, 86]'),
 Text(0.01416061067989575, 0.923728813559322, 'X[324] <= 16.5\ngini = 0.085\nsamples
= 1220\nvalue = [1166, 54]'),
 Text(0.012825899694630402, 0.9067796610169492, 'X[369] <= 56599.201\ngini =
0.081\nsamples = 1213\nvalue = [1162, 51]'),
 Text(0.012461337398308259, 0.8898305084745762, 'X[369] <= 13147.11\ngini =
0.079\nsamples = 1212\nvalue = [1162, 50]'),
 Text(0.01015647772409971, 0.8728813559322034, 'X[369] <= 12186.825\ngini =
0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.009791915427777565, 0.8559322033898306, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.010521040020421852, 0.8559322033898306, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.014766197072516809, 0.8728813559322034, 'X[0] <= 22637.0\ngini =
0.078\nsamples = 1210\nvalue = [1161, 49]'),
 Text(0.01125016461306614, 0.8559322033898306, 'X[369] <= 53936.924\ngini =
0.01\nsamples = 195\nvalue = [194, 1]'),
 Text(0.010885602316743995, 0.8389830508474576, 'gini = 0.0\nsamples = 169\nvalue =
[169, 0]'),
 Text(0.011614726909388282, 0.8389830508474576, 'X[369] <= 54036.645\ngini =
0.074\nsamples = 26\nvalue = [25, 1]'),
 Text(0.01125016461306614, 0.8220338983050848, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.011979289205710425, 0.8220338983050848, 'gini = 0.0\nsamples = 25\nvalue =
[25, 0]'),
 Text(0.0182822953196748, 0.8559322033898306, 'X[0] <= 22682.5\ngini =
0.09\nsamples = 1015\nvalue = [967, 48]'),
 Text(0.01791766723564534, 0.8389830508474576, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.018646791828289624, 0.8389830508474576, 'X[2] <= 23.5\ngini = 0.088\nsamples
= 1014\nvalue = [967, 47]'),
 Text(0.012708413798354712, 0.8220338983050848, 'X[278] <= 1.5\ngini =
0.069\nsamples = 780\nvalue = [752, 28]'),
 Text(0.008646961965890834, 0.8050847457627118, 'X[0] <= 144882.0\ngini =
0.054\nsamples = 719\nvalue = [699, 20]'),
 Text(0.0048988058568288, 0.788135593220339, 'X[369] <= 37391.641\ngini =
0.048\nsamples = 688\nvalue = [671, 17]'),
 Text(0.0035544823891408965, 0.7711864406779662, 'X[369] <= 23716.11\ngini =
0.01\nsamples = 193\nvalue = [192, 1]'),
 Text(0.0031899200928187533, 0.7542372881355932, 'X[369] <= 23257.23\ngini =
0.069\nsamples = 28\nvalue = [27, 1]'),
 Text(0.00282535779649661, 0.7372881355932204, 'gini = 0.0\nsamples = 27\nvalue =
[27, 0]'),
 Text(0.0035544823891408965, 0.7372881355932204, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.00391904468546304, 0.7542372881355932, 'gini = 0.0\nsamples = 165\nvalue =
[165, 0]'),
 Text(0.006243129324516703, 0.7711864406779662, 'X[369] <= 37509.736\ngini =
0.063\nsamples = 495\nvalue = [479, 16]'),
```