LAB ASSIGNMENT -08

Develop a program to implement Decision Tree model and analyze the model using confusion matrix
"""

```
import pandas as pd
import numpy as np
credit_df = pd.read_csv('German Credit Data.csv' )
credit_df.info()

credit_df.head()

credit_df.status.value_counts()

X_features = list( credit_df.columns )
X_features.remove( 'status' )
X_features

encoded_credit_df = pd.get_dummies(credit_df[X_features],drop_first = True)

list(encoded_credit_df.columns)

encoded_credit_df[['checkin_acc_A12','checkin_acc_A13','checkin_acc_A14']].head(5)

import statsmodels.api as sm
Y = credit_df.status
X = sm.add_constant(encoded_credit_df)

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, Y,test_size = 0.3,random_state = 42 )

from sklearn.tree import DecisionTreeClassifier
clf_tree_entropy = DecisionTreeClassifier( criterion = 'entropy',max_depth = 3 )
clf_tree_entropy.fit( X_train, y_train )

from sklearn.tree import export_graphviz
import pydotplus as pdot
#import pydotplus as pdot.graphviz
from IPython.display import Image

# Export the tree into odt file
export_graphviz( clf_tree_entropy,feature_names = X_train.columns )
out_file = 'chd_tree_entropy.odt'

# Read the create the image file
chd_tree_graph = pdot.graphviz.graph_from_dot_file( 'chd_tree_entropy.odt')
chd_tree_graph.write_jpg ( 'chd_tree_entropy.png' )
# Render the png file
Image(filename='chd_tree_entropy.png')

import math
entropy_node_1 = - (491/700)*math.log2(491/700) - (209/700)*math.log2(209/700)
```

```
print(round( entropy_node_1, 2))

from sklearn import metrics
tree_predict = clf_tree_entropy.predict( X_test )
metrics.roc_auc_score( y_test, tree_predict )

y_pred_df=pd.DataFrame( {"actual": y_test,"predicted": clf_tree_entropy.predict(X_test)})

def draw_cm( actual, predicted ):
    ## Cret
    cm = metrics.confusion_matrix( actual, predicted)
    sn.heatmap(cm, annot=True, fmt='.2f',xticklabels = ['Bad credit', 'Good Credit'],yticklabels =
["Bad credit", "Good Credit"] )
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sn
# %matplotlib inline
draw_cm( y_pred_df.actual,y_pred_df.predicted )

print( metrics.classification_report( y_pred_df.actual,y_pred_df.predicted ) )
```