import numpy as np  
import pandas as pd  
  
""" Read the data file"  
df = pd.read\_csv("./data/student-mat.csv", sep=";")  
  
""" ML helpers """  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import GridSearchCV, cross\_val\_score  
  
from sklearn.pipeline import Pipeline  
from sklearn.feature\_selection import SelectKBest, chi2  
from sklearn.svm import LinearSVC   
  
""" Split Data into Training and Testing Sets """  
def split\_data(X, Y):  
 return train\_test\_split(X, Y, test\_size=0.2, random\_state=17)  
  
""" Confusion Matrix """  
def confuse(y\_true, y\_pred):  
 cm = confusion\_matrix(y\_true=y\_true, y\_pred=y\_pred)  
 # print("\nConfusion Matrix: \n", cm)  
 fpr(cm)  
 ffr(cm)  
  
""" False Pass Rate """  
def fpr(confusion\_matrix):  
 fp = confusion\_matrix[0][1]  
 tf = confusion\_matrix[0][0]  
 rate = float(fp) / (fp + tf)  
 print("False Pass Rate: ", rate)  
  
""" False Fail Rate """  
def ffr(confusion\_matrix):  
 ff = confusion\_matrix[1][0]  
 tp = confusion\_matrix[1][1]  
 rate = float(ff) / (ff + tp)  
 print("False Fail Rate: ", rate)  
  
 return rate  
  
""" Train Model and Print Score """  
def train\_and\_score(X, y):  
 X\_train, X\_test, y\_train, y\_test = split\_data(X, y)  
  
 clf = Pipeline([  
 ('reduce\_dim', SelectKBest(chi2, k=2)),  
 ('train', LinearSVC(C=100))  
 ])  
  
 scores = cross\_val\_score(clf, X\_train, y\_train, cv=5, n\_jobs=2)  
 print("Mean Model Accuracy:", np.array(scores).mean())  
  
 clf.fit(X\_train, y\_train)  
  
 confuse(y\_test, clf.predict(X\_test))  
 print()  
  
""" Main Program """  
def main():  
 print("\nStudent Performance Prediction")  
  
 # For each feature, encode to categorical values  
 class\_le = LabelEncoder()  
 for column in df[["school", "sex", "address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "guardian", "schoolsup", "famsup", "paid", "activities", "nursery", "higher", "internet", "romantic"]].columns:  
 df[column] = class\_le.fit\_transform(df[column].values)  
  
 # Encode G1, G2, G3 as pass or fail binary values  
 for i, row in df.iterrows():  
 if row["G1"] >= 10:  
 df["G1"][i] = 1  
 else:  
 df["G1"][i] = 0  
  
 if row["G2"] >= 10:  
 df["G2"][i] = 1  
 else:  
 df["G2"][i] = 0  
  
 if row["G3"] >= 10:  
 df["G3"][i] = 1  
 else:  
 df["G3"][i] = 0  
  
 # Target values are G3  
 y = df.pop("G3")  
  
 # Feature set is remaining features  
 X = df  
  
 print("\n\nModel Accuracy Knowing G1 & G2 Scores")  
 print("=====================================")  
 train\_and\_score(X, y)  
  
 # Remove grade report 2  
 X.drop(["G2"], axis = 1, inplace=True)  
 print("\n\nModel Accuracy Knowing Only G1 Score")  
 print("=====================================")  
 train\_and\_score(X, y)  
  
 # Remove grade report 1  
 X.drop(["G1"], axis=1, inplace=True)  
 print("\n\nModel Accuracy Without Knowing Scores")  
 print("=====================================")  
 train\_and\_score(X, y)