1. **Xử lý dữ liệu**

import numpy as np **# linear algebra**

import pandas as pd **# data processing, CSV file I/O (e.g. pd.read\_csv)**

import matplotlib.pyplot as plt **# for data visualization purposes**

import seaborn as sns **# for statistical data visualization**

%matplotlib inline

**# import thư viện để hiển thị các warning nếu như có warning xảy ra**

import warnings

warnings.filterwarnings('ignore')

**# Đọc dữ liệu từ file adult.csv. Lưu ý trước khi đọc thì cần xem dữ liệu có điểm gì đặc biệt**

data = 'adult.csv'

df = pd.read\_csv(data, header=None, sep=',\s')

**# Hiển thị số dòng số cột của dữ liệu vừa đọc**

df.shape

**# Hiển thị dataset**

df.head()

**# Khởi tạo tên cột cho bộ dữ liệu**

col\_names = ['age', 'workclass', 'fnlwgt', 'education', 'education\_num', 'marital\_status', 'occupation', 'relationship', 'race', 'sex', 'capital\_gain', 'capital\_loss', 'hours\_per\_week', 'native\_country', 'income']

df.columns = col\_names

df.head()

**# Tìm các cột có kiểu dữ liệu không phải kiểu số**

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

df[categorical].head()

**# Đếm các chỗ bị trống ở các cột có kiểu dữ liệu là object**

df[categorical].isnull().sum()

**# Kiểm tra các nhãn trong cột workclass phân bố như thế nào**

df.workclass.value\_counts()

**# Loại bỏ giá trị ? trong cột workclass thành NaN**

df['workclass'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột workclass**

df.workclass.value\_counts()

**# Kiểm tra sự phân bộ trên các nhãn trong cột workclass**

df.education.value\_counts()

**# Loại bỏ giá trị ? trong cột education thành NaN**

df['education'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột education**

df.education.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột marital\_status**

df.marital\_status.value\_counts()

**# Loại bỏ giá trị ? trong cột marital\_status thành NaN**

df['marital\_status'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột marital\_status**

df.marital\_status.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột occupation**

df.occupation.value\_counts()

**# Loại bỏ giá trị ? trong cột occupation thành NaN**

df['occupation'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột occupation**

df.occupation.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột relationship**

df.relationship.value\_counts()

**# Loại bỏ giá trị ? trong cột relationship thành NaN**

df['relationship'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột relationship**

df.relationship.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột race**

df.race.value\_counts()

**# Loại bỏ giá trị ? trong cột race thành NaN**

df['race'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột race**

df.race.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột sex**

df.sex.value\_counts()

**# Loại bỏ giá trị ? trong cột sex thành NaN**

df['sex'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột sex**

df.sex.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột native\_country**

df.native\_country.value\_counts()

**# Loại bỏ giá trị ? trong cột native\_country thành NaN**

df['native\_country'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột native\_country**

df.native\_country.value\_counts()

**# Kiểm tra sự phân bố trên các nhãn trong cột income**

df.income.value\_counts()

**# Loại bỏ giá trị ? trong cột income thành NaN**

df['income'].replace('?', np.NaN, inplace=True)

**# Kiểm tra lại sự phân bố của các nhãn trong cột income**

df.income.value\_counts()

**# Tìm kiếm các cột dữ liệu có kiểu số**

numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :', numerical)

**# Hiển thị các cột có kiểu dữ liệu số**

df[numerical].head()

**# Kiểm tra các giá trị bị thiếu trong các cột có kiểu dữ liệu số**

df[numerical].isnull().sum()

**# Chọn bộ X và bộ y để training**

X = df.drop(['income'], axis=1)

y = df['income']

**# Phân chia bộ X và y thành bộ training và bộ test**

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 = 0)

**# Kiểm tra kích thước của bộ X\_train và X\_test**

X\_train.shape, X\_test.shape

**# Kiểm tra kiểu dữ liệu của các cột trên bộ X\_train**

X\_train.dtypes

**# Hiển thị các cột của bộ X\_train có kiểu dữ liệu là object**

categorical = [col for col in X\_train.columns if X\_train[col].dtypes == 'O']

categorical

**# Hiển thị các cột của bộ X\_train có kiểu dữ liệu kiểu số**

numerical = [col for col in X\_train.columns if X\_train[col].dtypes != 'O']

numerical

**# Tính % các giá trị bị thiếu trong các cột dữ liệu kiểu object trong bộ X\_training**

X\_train[categorical].isnull().mean()

**# Tìm ra các cột có chứa giá trị bị thiếu trong các cột có kiểu dữ liệu object trong bộ #X\_training**

for col in categorical:

if X\_train[col].isnull().mean()>0:

print(col, (X\_train[col].isnull().mean()))

**# Tìm các giá trị xuất hiện nhiều nhất trong các cột và gán giá trị đó cho các giá trị bị thiếu**

for df2 in [X\_train, X\_test]:

df2['workclass'].fillna(X\_train['workclass'].mode()[0], inplace=True)

df2['occupation'].fillna(X\_train['occupation'].mode()[0], inplace=True)

df2['native\_country'].fillna(X\_train['native\_country'].mode()[0], inplace=True)

**# Kiểm tra lại các cột có kiểu dữ liệu object trông bộ X\_train**

X\_train[categorical].isnull().sum()

**# Kiểm tra giá trị bị thiếu trong các cột có kiểu dữ liệu object trong bộ X\_test**

X\_test[categorical].isnull().sum()

**# Kiểm tra giá trị còn thiếu trong bộ X\_train**

X\_train.isnull().sum()

**# import category encoders**

import category\_encoders as ce

**# Mã hoá dữ liệu bằng One-hot encoding**

encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital\_status', 'occupation', 'relationship',

'race', 'sex', 'native\_country'])

X\_train = encoder.fit\_transform(X\_train)

X\_test = encoder.transform(X\_test)

cols = X\_train.columns

from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

X\_train = pd.DataFrame(X\_train, columns=[cols])

X\_test = pd.DataFrame(X\_test, columns=[cols])

1. **Xây dựng mô hình**

**# Huấn luyện bằng Gaussian Naive Bayes classifier trên bộ traning**

from sklearn.naive\_bayes import GaussianNB

**# khởi tạo mô hình**

gnb = GaussianNB()

**# Khớp mô hình với các bộ cần training**

gnb.fit(X\_train, y\_train)

y\_pred= gnb.predict(X\_test)

**# Tính toán hiệu suất với chỉ số accuracy\_score**

from sklearn.metrics import accuracy\_score

print('Model accuracy score: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

**# Dự đoán kết quả từ bộ X\_train**

y\_pred\_train = gnb.predict(X\_train)

y\_pred\_train

**# In ra điểm hiệu suất của từng bộ training và bộ test dựa vào mô hình GaussianNB**

print('Training set score: {:.4f}'.format(gnb.score(X\_train, y\_train)))

print('Test set score: {:.4f}'.format(gnb.score(X\_test, y\_test)))

**# Kiểm tra phân bố dữ liệu trên bộ test**

y\_test.value\_counts()

**# Xây dự ma trận nhầm lẫn**

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])

**# Hiển thị ma trận nhầm lẫn**

cm\_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],

index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm\_matrix, annot=True, fmt='d', cmap='YlGnBu')

**# Tìm ra 10 xác suất đự đoán đầu tiên**

y\_pred\_prob = gnb.predict\_proba(X\_test)[0:10]

y\_pred\_prob

**# Lưu trữ giá trị dữ đoán cho lớp 1 với giá trị > 50K**

y\_pred1 = gnb.predict\_proba(X\_test)[:, 1]

**# Vẽ biểu đồ khả năng dự đoán của mô hình**

plt.rcParams['font.size'] = 12

plt.hist(y\_pred1, bins = 10)

plt.title('Histogram of predicted probabilities of salaries >50K')

plt.xlim(0,1)

plt.xlabel('Predicted probabilities of salaries >50K')

plt.ylabel('Frequency')