ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG

BÁO CÁO PROJECT 2

| ĐỀ TÀI |

CLASSIFICATION

| GIÁO VIÊN HƯỚNG DẪN |

Thầy Lê Ngọc Thành

Thầy Nguyễn Thái Vũ

Thành phố Hồ Chí Minh

THÀNH VIÊN NHÓM

MÃ SỐ	HỌ VÀ TÊN
19127472	Nguyễn Bá Minh
19127481	Trần Hoàng Nam
19127595	Nguyễn Minh Trí

I. MỨC ĐỘ HOÀN THÀNH

- Xây dựng mô hình phân lớp bằng mô hình random forest
- Xây dựng ứng dụng để trực quan hoá và huấn luyện mô hình random forest

⇨Đánh giá: Hoàn thành 100%

II. XÂY DỰNG MÔ HÌNH PHÂN LỚP

- Dữ liệu được lấy từ [Breast Cancer Wisconsin (Diagnostic) Data Set | Kaggle]
- Chúng ta tiến hành đọc file

```
data = pd.read_csv('../Data/data.csv')
data.head()
```

✓ 0.1s Python

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 33 columns

1. Tiền xử lý

Đầu tiên, chúng ta cần xem xét dữ liệu có có bị rỗng hay không

```
a = data.isnull().sum()
a[a != 0]
```

✓ 0.6s

```
Unnamed: 32    569
dtype: int64
```

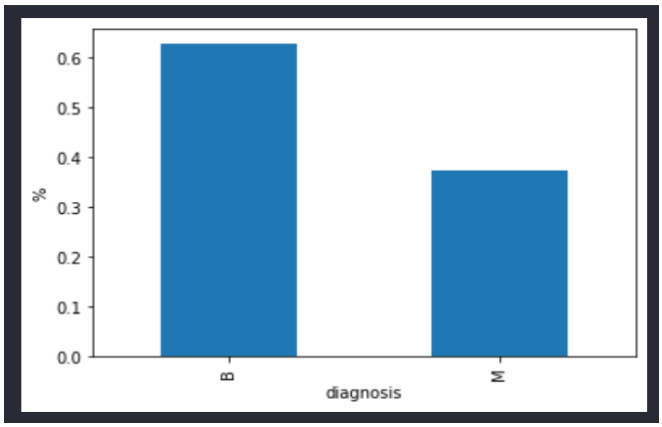
Có vẻ như file có một chút vấn đề nhỏ nên đã đọc dư một cột unnamed: 32

Tiếp theo, chúng ta xem xét đến kiểu dữ liệu của các trường dữ liệu từ file đọc vào

```
id                         int64
diagnosis                  object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

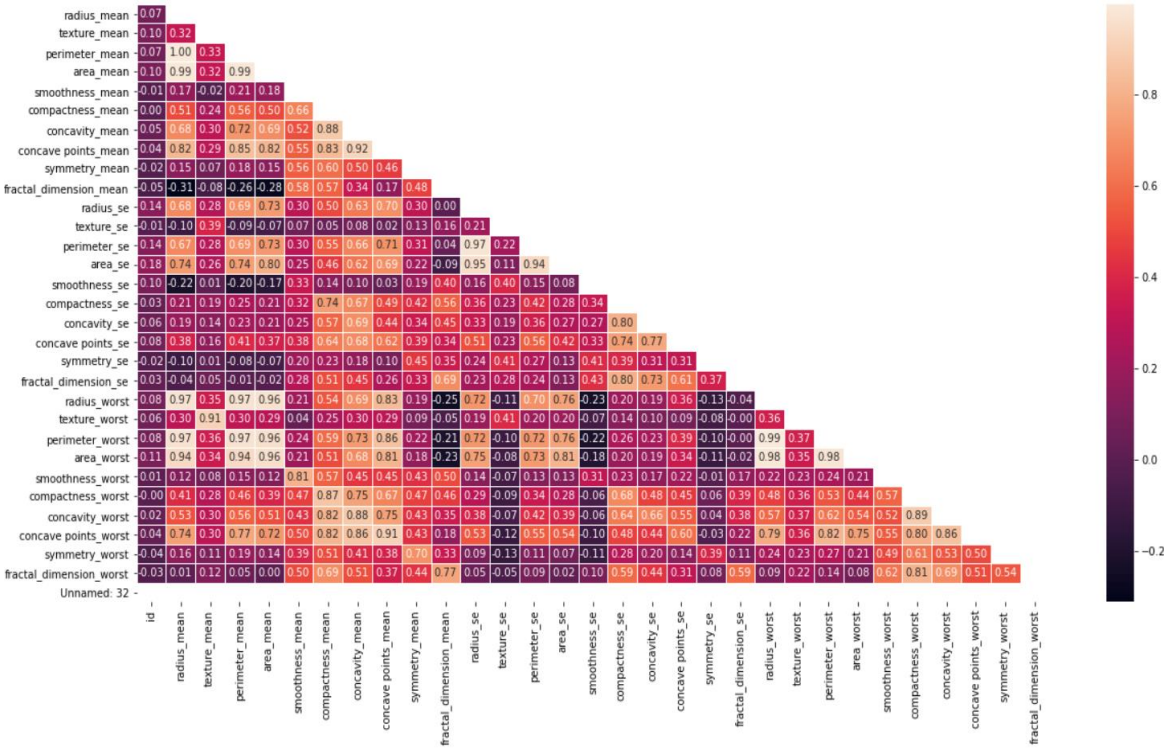
- Có thể thấy rằng tất cả các cột dữ liệu của chúng ta hầu hết là numeric – ngoại trừ cột kết quả diagnosis là lớp nhị phân
- Chúng ta để ý rằng, cột id là độc nhất, do đó mà cột này không cần thiết cho quá trình phân lớp, vì thế chúng ta sẽ loại bỏ cột id và cột Unnamed: 32(cột này xuất hiện là do vấn đề đọc file như đã giải thích ở trên)

Chúng ta sẽ xem xét tỉ lệ của lớp nhị phân diagnosis



⇨Tuy lớp B có nhiều hơn lớp M, nhưng nhìn chung thì sự chênh lệch này vẫn có thể chấp nhận được

Tiếp theo, chúng ta sẽ tiến hành xem xét sự tương quan giữa các cột dữ liệu



Nhóm có một vài nhận xét về biểu đồ này:

- Có nhiều cặp biến có độ tương quan rất cao. Có thể thấy những cặp biến có độ tương quan > 0.9. VD: **perimeter_mean** và **perimeter_worst** có độ tương quan 0.97, để tránh việc overfitting thì chúng ta sẽ loại bỏ các cột này
- Dữ liệu sau khi loại bỏ các cột dư thừa, chúng ta giảm từ 33 cột xuống còn 21 cột
- Chúng ta sẽ lưu dư liệu này vào file mới là preprocess_data.csv để cho ứng dụng sử dụng sau này

2. Phân lớp

- Trước khi chạy phân lớp dữ liệu, chúng ta sẽ chia data thành 2 tập train và test

```
from sklearn.model_selection import train_test_split
X = data.drop('diagnosis', axis=1)
y = data['diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

✓ 0.2s

Tiến hành chạy mô hình phân lớp nhị phân, ở đây, chúng ta sẽ dùng Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

lr_model = DecisionTreeClassifier()
lr_model.fit(X_train, y_train)
```

✓ 0.1s

Sau đó, tiến hành kiểm tra và thu được kết quả

```
Độ chính xác: (Trên tập huấn luyện) 1.00
Độ chính xác: (Trên tập kiểm thử) 0.92
```

	precision	recall	f1-score	support
B	0.98	0.88	0.93	67
M	0.85	0.98	0.91	47
accuracy			0.92	114
macro avg	0.92	0.93	0.92	114
weighted avg	0.93	0.92	0.92	114

Nhìn sơ qua, chúng ta thấy rằng việc sử dụng một mô hình đơn giản Decision Tree classifier vẫn cho độ chính xác khá cao, tất cả các chỉ số **precision**, **recall**, **f1-score** đều cao. Độ chính xác của tập test đạt 92%
Nhìn chung thì tập dữ liệu này vẫn chưa đủ lớn, chúng ta sẽ dùng K-fold cross validation để đánh giá mô hình trên đầy đủ và chính xác hơn với tập dữ liệu nhỏ

- Chúng ta sẽ cho tham số n_splits = splits =3

```
kfold = KFold(n_splits=splits)
```

- Sau đó tiến hành đánh giá kết quả phân lớp

```
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.94
Đã train xong Fold  1
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.93
Đã train xong Fold  2
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.90
Đã train xong Fold  3
Độ chính xác trung bình sau 3 lần là: 0.92
```

- Độ chính xác trung bình sau 3 lần test là 92%, chúng em sẽ tiếp tục sử dụng mô hình khác là mô hình KNeighbors và thu được kết quả

```
Độ chính xác mô hình trên tập train: 0.95
Độ chính xác mô hình trên tập validation: 0.86
Đã train xong Fold  1
Độ chính xác mô hình trên tập train: 0.93
Độ chính xác mô hình trên tập validation: 0.92
Đã train xong Fold  2
Độ chính xác mô hình trên tập train: 0.93
Độ chính xác mô hình trên tập validation: 0.89
Đã train xong Fold  3
Độ chính xác trung bình sau 3 lần là: 0.89
```

- Có vẻ như mô hình KNN cho ra kết quả khá tệ, thấp hơn so với decision tree

- Tiếp tục thử nghiệm với mô hình Logistic Regression

```
Độ chính xác mô hình trên tập train: 0.96
Độ chính xác mô hình trên tập validation: 0.92
Đã train xong Fold  1
Độ chính xác mô hình trên tập train: 0.94
Độ chính xác mô hình trên tập validation: 0.97
Đã train xong Fold  2
Độ chính xác mô hình trên tập train: 0.96
Độ chính xác mô hình trên tập validation: 0.94
Đã train xong Fold  3
Độ chính xác trung bình sau 3 lần là: 0.94
```

- Kết quả thu được khá tốt, cao hơn 2 mô hình trước(94%), ở fold 2 độ chính xác đạt tới tận 98%
- Tiếp tục thử nghiệm với mô hình Random Forest, vì mô hình này tổng hợp nhiều cây nên chúng ta hy vọng sẽ có kết quả tốt hơn

```
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.93
Đã train xong Fold  1
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.98
Đã train xong Fold  2
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.95
Đã train xong Fold  3
Độ chính xác trung bình sau 3 lần là: 0.95
```

Độ chính xác trung bình đã tăng lên 95%, ở tập thứ 2 còn tăng lên đến tận 98%, có vẻ đây là một mô hình tiềm năng, chúng em sẽ tiến hành dùng **GridSearch** để tìm siêu tham số cho mô hình này(Random forest classifier)

Đây là kết quả tham số thu được:

```
{'criterion': 'entropy',
 'max_depth': 13,
 'min_samples_leaf': 3,
 'min_samples_split': 2,
 'n_estimators': 100}
```

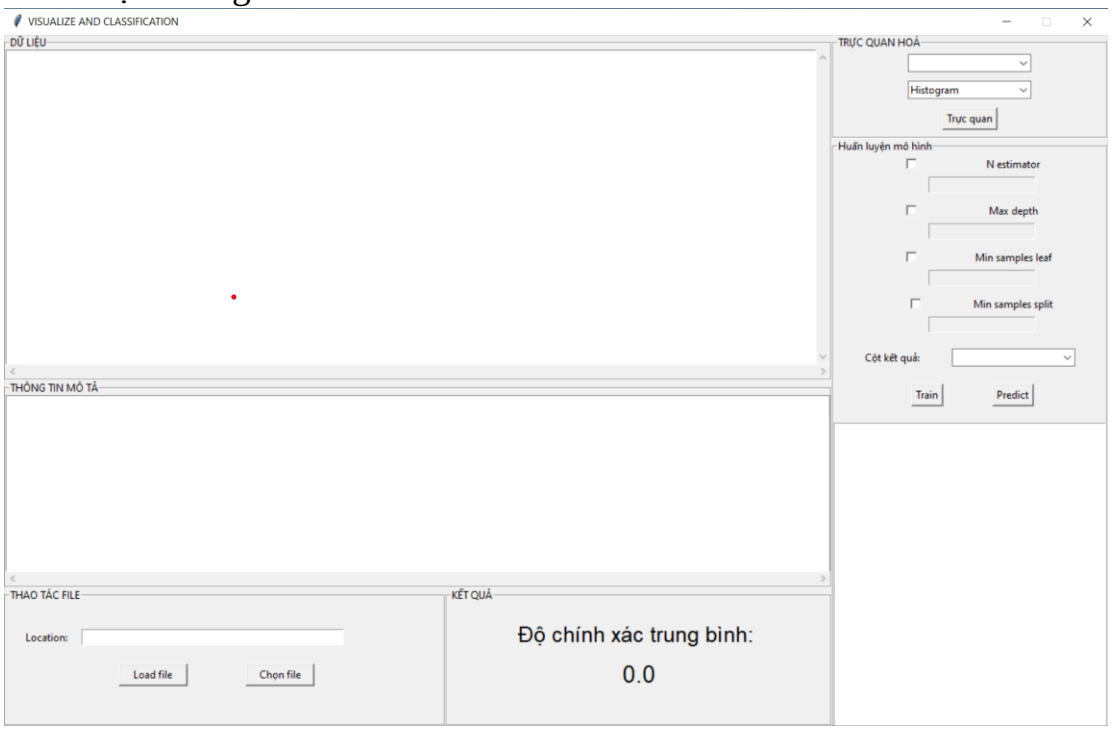
Tiến hành sử dụng các tham số này và đánh giá

```
Độ chính xác mô hình trên tập train: 0.99
Độ chính xác mô hình trên tập validation: 0.92
Đã train xong Fold  1
Độ chính xác mô hình trên tập train: 0.99
Độ chính xác mô hình trên tập validation: 0.98
Đã train xong Fold  2
Độ chính xác mô hình trên tập train: 1.00
Độ chính xác mô hình trên tập validation: 0.97
Đã train xong Fold  3
Độ chính xác trung bình sau 3 lần là: 0.96
```

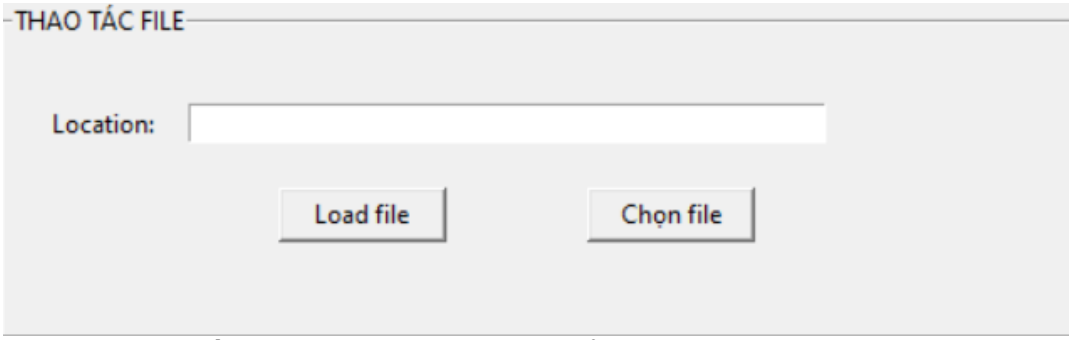
Nhân thấy mô hình đã thoả đạt độ chính xác lên đến 96%, nhóm quyết định chọn mô hình random forest classifier để làm

III. ỨNG DỤNG

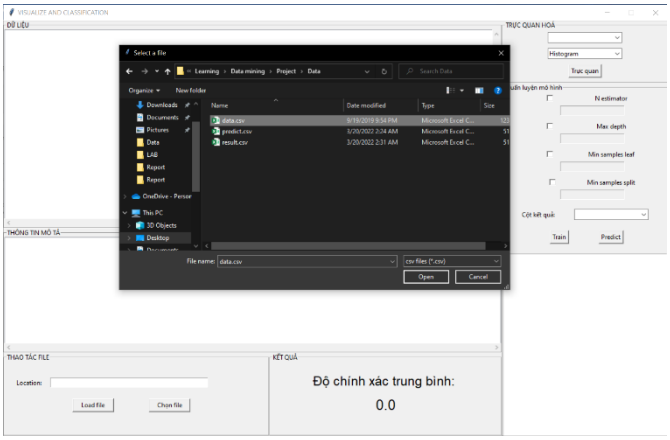
Giao diện chung



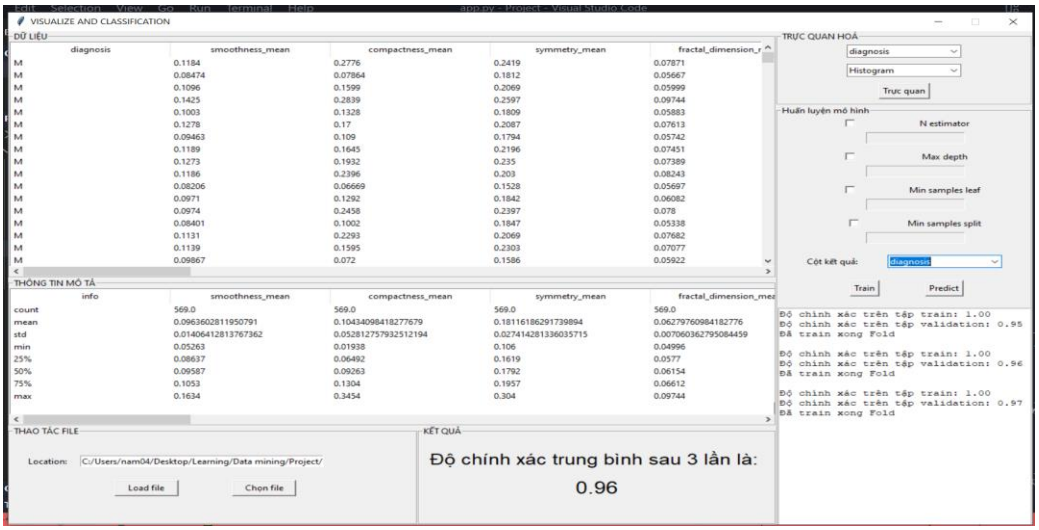
Khu vực THAO TÁC FILE là khu vực đầu tiên người dùng cần quan tâm đến



Chúng ta sẽ bấm vào nút “Chọn file” để chọn file dữ liệu

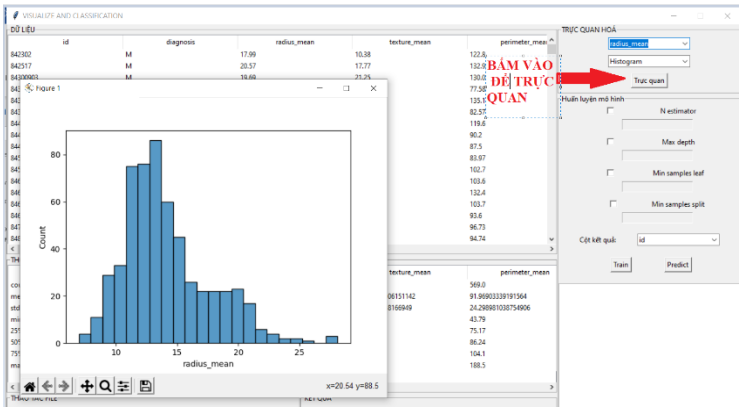


- Sau khi đã chọn được file dữ liệu thì bấm “Load file” để file được load vào chương trình

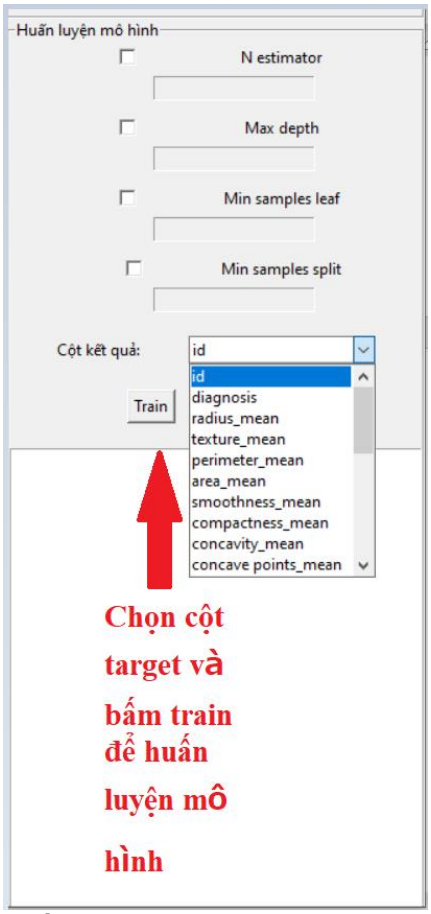


- Phần “DỮ LIỆU” hiển thị dữ liệu được load lên chương trình
- Phần “THÔNG TIN MÔ TẢ” thể hiện các thông tin chung về các trường dữ liệu số

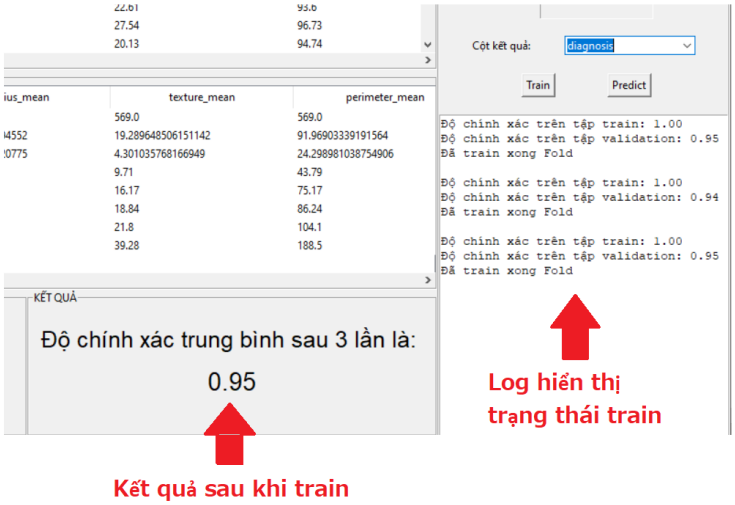
- Drop down thứ 1 của phần “TRỰC QUAN HOÁ” ở trên dùng để chọn cột để trực quan hoá, hiện tại chương trình chỉ cho phép trực quan hoá 1 cột duy nhất
- Drop down thứ 2 của phần “TRỰC QUAN HOÁ” cho phép chọn loại biểu đồ, hiện tại chỉ có 3 loại biểu đồ là histogram, pie chart và bar chart



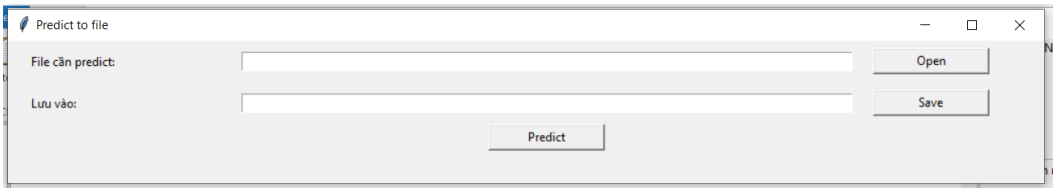
- Phần “HUẤN LUYỆN MÔ HÌNH”, chúng ta có các checkbox, khi tick vào các checkbox thì có thể edit tham số cho mô hình Random Forest Classifier của chúng ta
- Cột kết quả là cột target(y) mà chúng ta muốn dự đoán



- Tiến hành chọn cột target và điều chỉnh thông số theo ý muốn, bấm train ta thu được kết quả

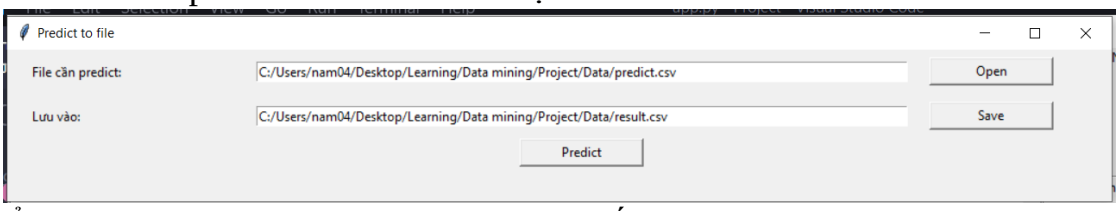


- Sau khi train, chúng ta có thể bấm nút “Predict” để sử dụng mô hình phân lớp này

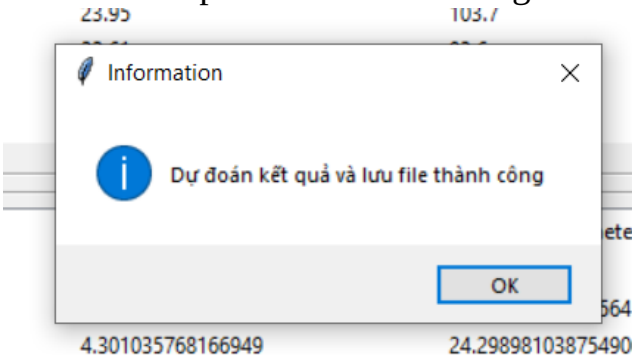


Ở đây, chúng ta sẽ load file cần phân lớp bằng cách nhập đường dẫn hoặc duyệt file bằng nút Open

- Nút Save thì sẽ chọn đường dẫn để lưu file kết quả
- Sau đó bấm predict thì sẽ file sẽ được xuất ra



- Ở đây, nhóm đã tạo file dữ liệu predict vốn là một sample của file data.csv nhưng không có cột target, mô hình sẽ tiến hành phân lớp cho dữ liệu từ file target này và xuất ra file result.csv
- Sau khi bấm predict thì sẽ có thông báo



- Lưu ý file predict được tạo ra từ file generate_predict_data.ipynb

IV. THAM KHẢO

How to view Excel File or Pandas DataFrame in Tkinter (Python GUI) - YouTube