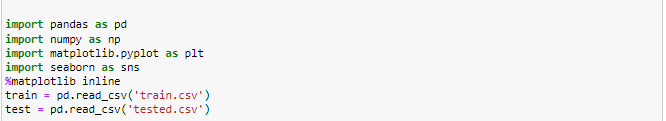
1. Chạy kết hợp để so sánh các phương pháp

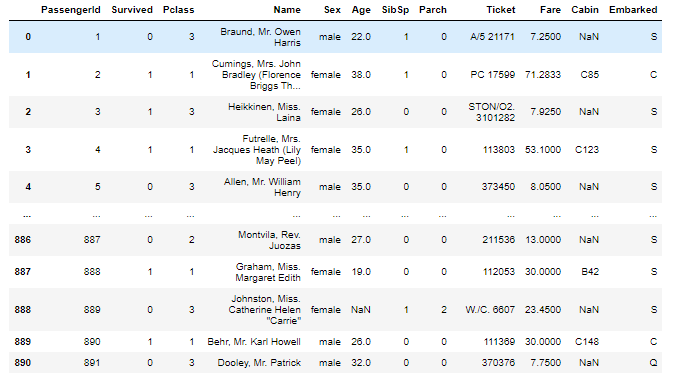
- Về dữ liệu, nhóm chúng em sẽ sử dụng 2 bộ dữ liệu và titanic và banking với 2 cách tiền xử lý và làm sạch dữ liệu khác nhau nhắm khẳng định sự đa dạng trong cách xử lí dữ liệu

- Đầu tiên, em sẽ tiến hành trên bộ dữ liệu về titanic



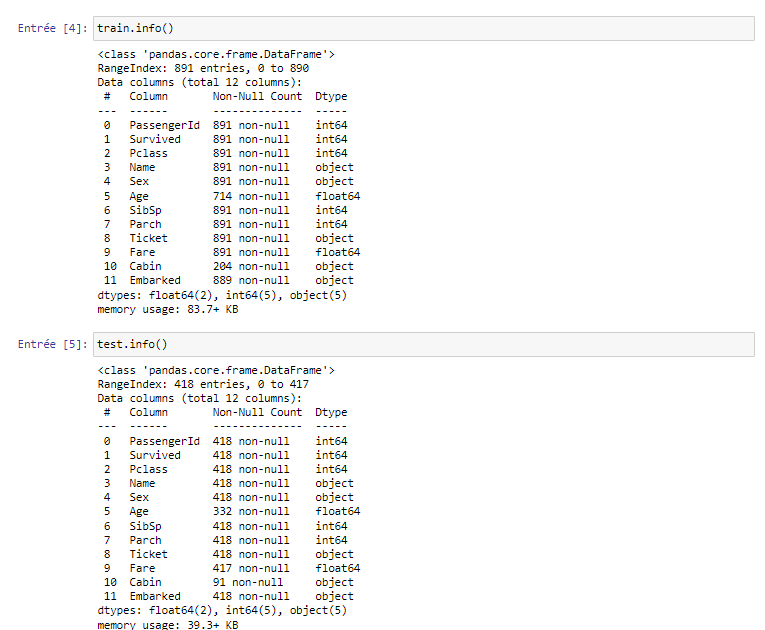
- Ở đây bộ dữ liệu của bọn em đã được chia thành 2 file train và test và bọn e sẽ làm sạch dữ liệu trên cả 2 file đó

- Dưới đây là bộ dữ liệu titanic



- Trong đó PassengerId là ID của khách trên tàu, Survived là họ còn sống hay không sau vụ chìm tàu, Pclass là hạng của vé, Sex là giới tính, sipsp là có bao nhiêu người thân kiểu dạng anh, chị, em, vợ, chồng đi cùng trên tàu, Parch là có bao nhiêu người thân kiểu dạng bố mẹ, con trai, con gái đi cùng trên tàu, Ticket là số vé, Fare là phí đi tàu, Cabin là số khoang và embarked là lên tàu ở đâu (với C là Cherbouurg, Q là Queenstown và S là Southamton)

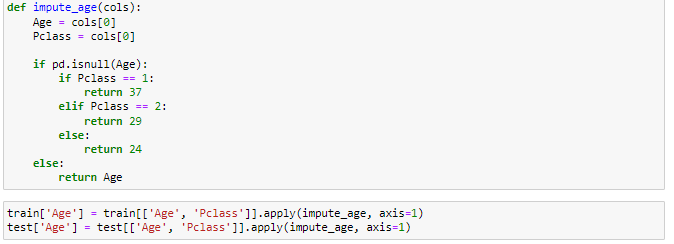
- Tiếp theo, em sẽ xem các thống tin về 2 dataframe train và test



- Ở đây chúng em nhận thấy phần khoang Cabin có dữ liệu bị khuyết rất lớn ở cả 2 bảng nên bọn e sẽ tiến hành xóa cột đó ra khỏi bảng

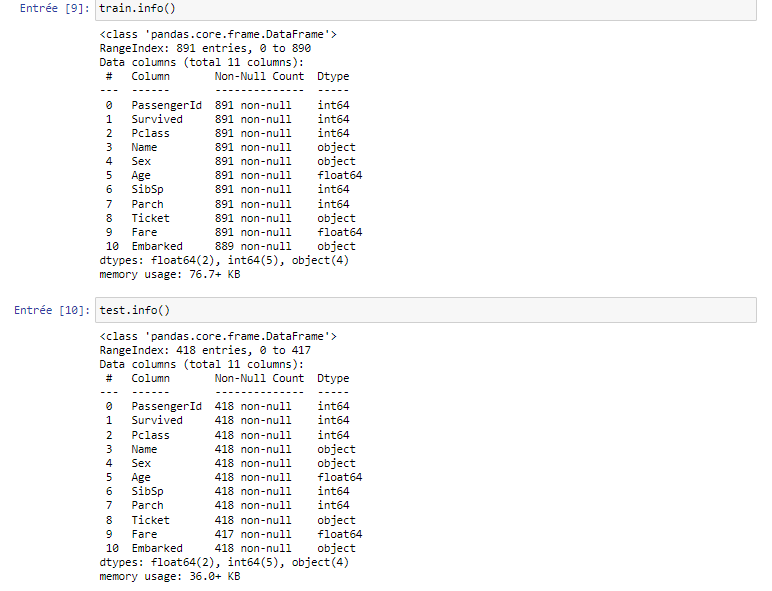


- Tiếp theo bọn em sẽ tiến hành xử lí các cột về tuổi bị mang giá trị null ở cả hai bảng bằng hạng vé trên tàu

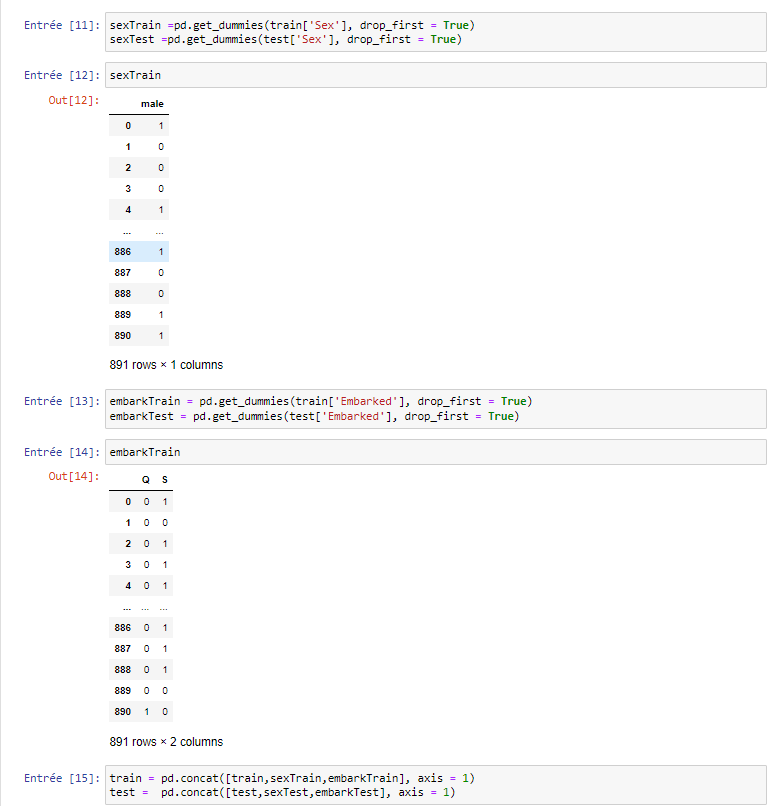


- Ở đây bọn em sẽ mặc định những người ở hạng 1 sẽ là 37 tuổi (vì hạng này cấp cao nhất nên sẽ thường là những người thuộc tầng lớp trung niên), với hạng 2 là 29 tuổi và còn lại là 24 tuổi. Và áp dụng luôn vào 2 frame đang xử lí

- Tiếp tục 1 lần nữa kiểm tra lại info của 2 dataframe



- Và để dễ dàng cho quá trình đưa cho máy học bộ dữ liệu thì ta nên chuyển hết tất cả các dữ liệu về dạng số. Do đó bọn em dùng hàm get\_dummies trong python để đưa các cột sex và embarked về dạng số và tiến hành nối vào bảng như sau

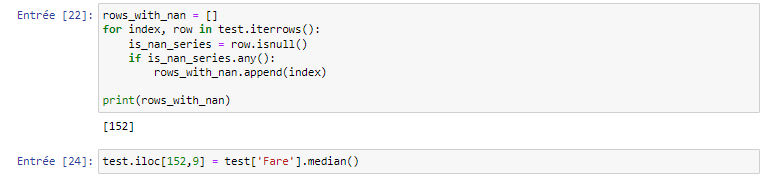


- Tiếp theo bọn em tiến hành xử lí cột tên khách hàng như sau

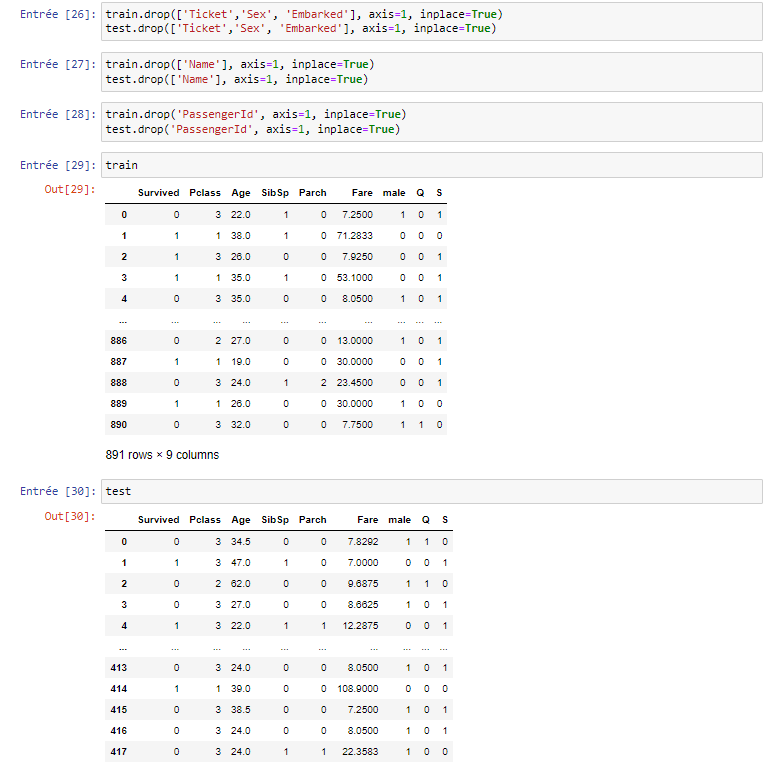


- Chúng em sẽ chia các tên của khách hàng theo tiền tố và biến nó thành số với quy ước trong hàm impute\_name rồi nối vào bảng

- Tiếp theo, nhận thấy cột Fare bên bảng test bị thiếu 1 giá trị nên chúng em đã gán nó bằng trung vị của cột Fare



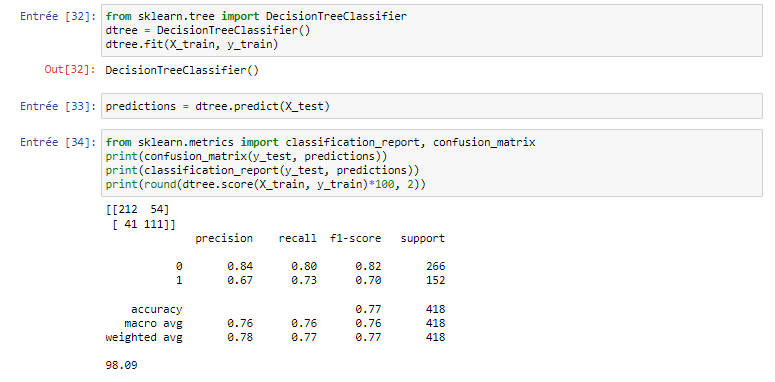
- Sau đó em tiến hành xóa tất cả các cột chữ trong bảng và xem lại bảng 1 lần nữa



- Tiếp theo, ta chia bộ dữ liệu thành X\_train, y\_train, X\_test, y\_test để tiến hành train mô hình và chạy thử

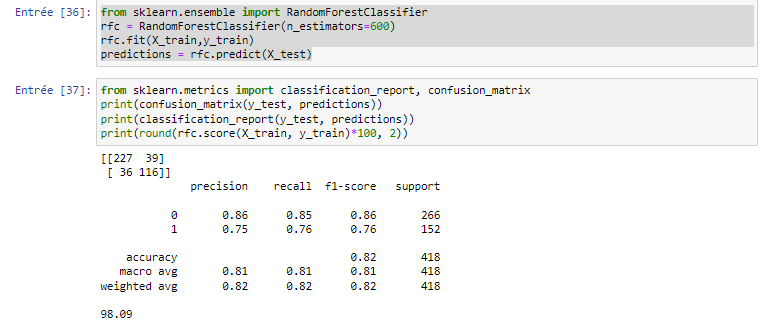


- Đầu tiên em sẽ tiến hành chạy thử với model Decision Tree



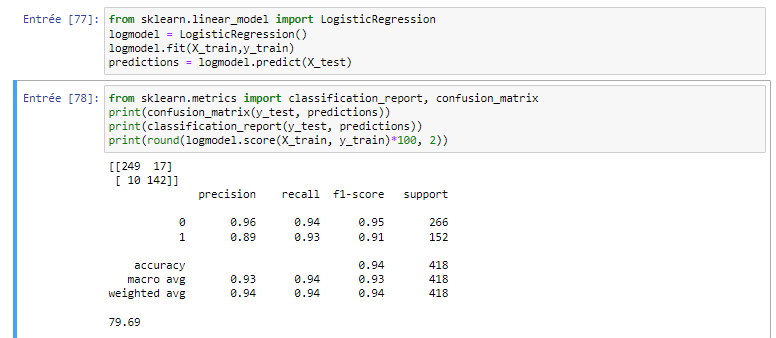
- Ta nhận thấy độ chính xác của cây quyết định với dataset này là 77% với confusion matrix ở góc trên của output

- Ta tiến hành chạy với phương pháp Random Forest với 600 cây



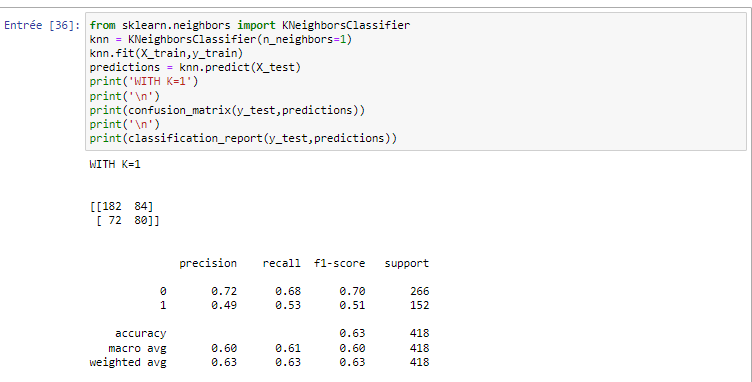
- Độ chính xác ở đây đã lên 82%

- Ta tiến hành chạy thử với phương pháp LogisticRegression



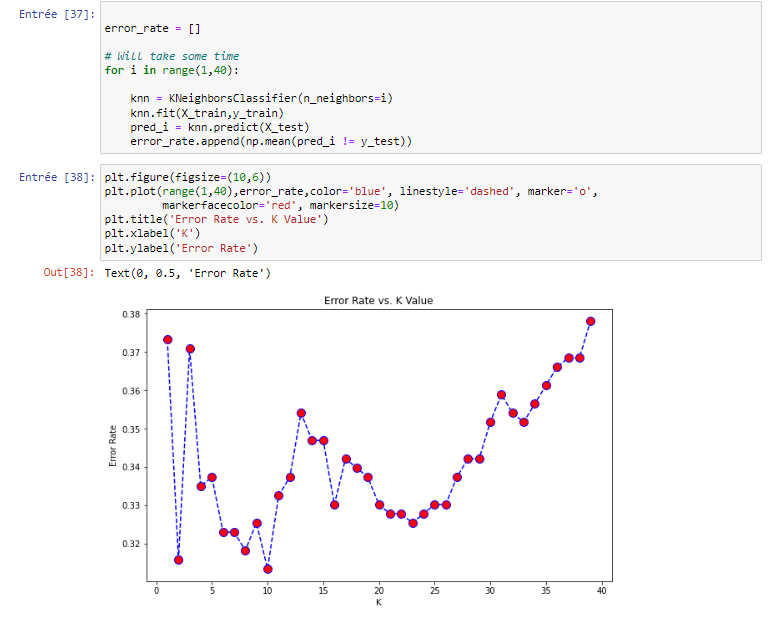
- Ở đây ta thấy độ chính xác của thuật toán lên tới 94%

- Tiếp theo là thuật toán KNN với K=1

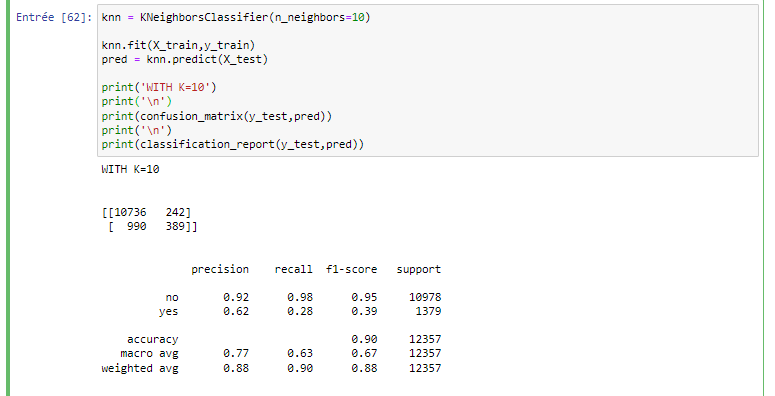


- Ở đây ta thấy độ chính xác chỉ có 63%

- Ta tiến hành tìm giá trị K sao cho thuật toán có tỉ lệ lỗi thấp nhất (K chạy từ 1 đến 40)

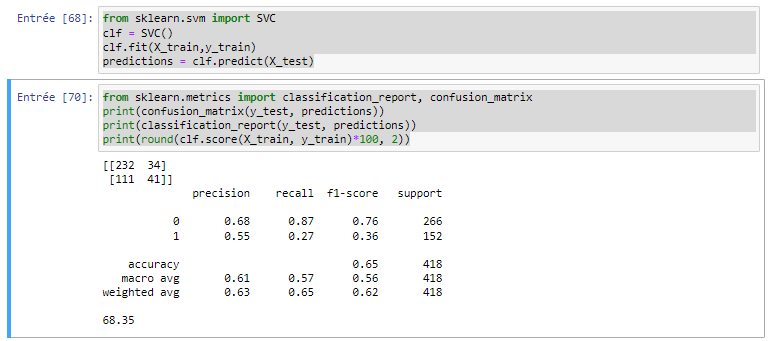


- Ta nhận thấy ở K=10, mô hình có tỉ lệ lỗi thấp nhất, chạy mô hình với giá trị K này



- Ta nhận thấy tỉ lệ chính xác đã lên 69%

- Cuối cùng là mô hình SVC



- Mô hình này chỉ có độ chính xác là 65%

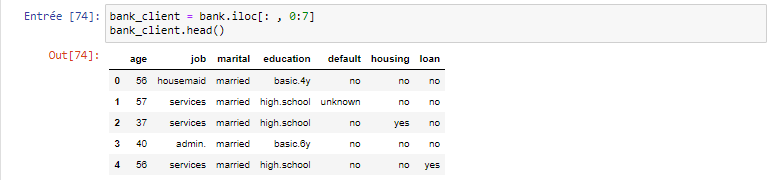
=> Từ đó ta kết luân được phương pháp LogisticReression và Decision tree sẽ phù hợp hơn các phương pháp còn lại để giải quyết các dataset có dạng như Titanic

- Tiếp theo ta sẽ sử dụng data liên quan đến ngân hàng tài chính, bảng sẽ có 21 cột và hơn 41188 dòng về data từ các ngân hàng thu thập được về khách hàng của họ, các thông tin được đưa vào sẽ rất đa dạng bao gồm tuổi, nghề nghiệp, tình trạng hôn nhân, trình độ học vấn, thuê nhà, thời gian gần nhất kí hợp đồng giao dịch, và các thuộc tính khác

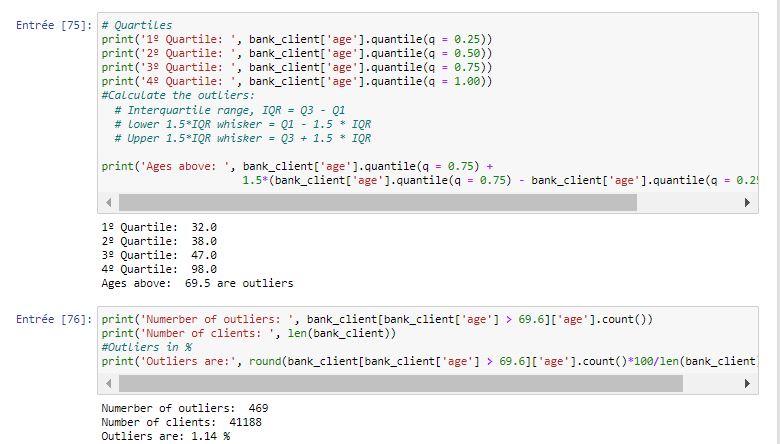


- Với data này, chúng em sẽ tách bảng thành nhiều phần khác nhau để tiến hành tiền xử lí dữ liệu

- Đầu tiên, ta sẽ tách những cột liên quan đến thông tin cá nhân

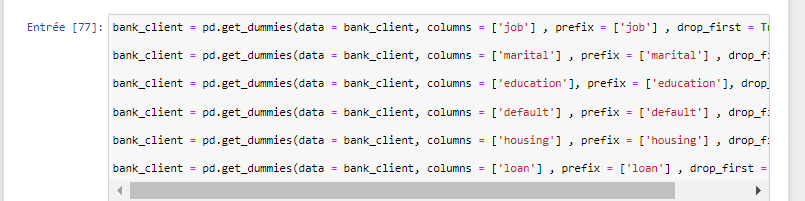


- Sau đó, chúng em sẽ tiến hành tính khoảng tứ phân vị cho cột tuổi để tìm được các giá trị nhiễu (outliers)

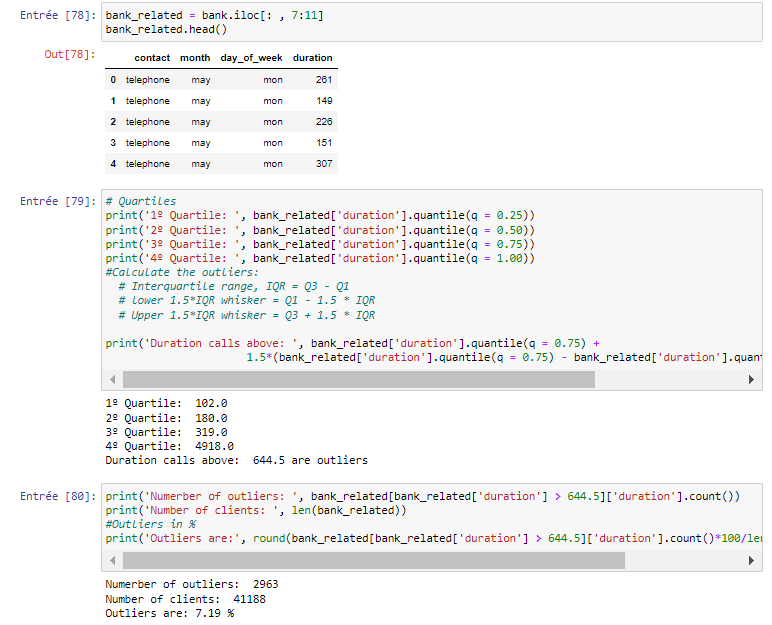


- Vì những giá trị này chiếm tỉ lệ ít nên ở đây bọn em sẽ giữ nguyên.

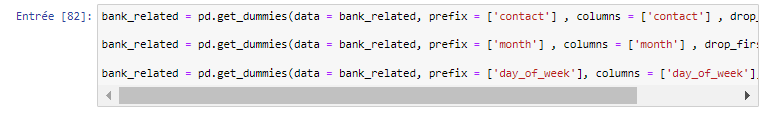
- Tiếp theo, chúng em sẽ sử dụng hàm get\_dummies đối với các cột như data Titanic ở trên



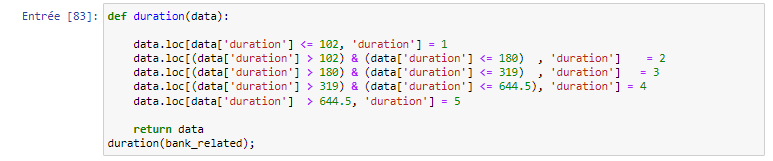
- Tiếp theo chúng em lại tách những thành phần liên quan trực tiếp tới ngân hàng và sử lí giống với phần trên



- Tiếp đến bọn em tiếp tục dùng hàm get\_dummies với các cột chưa có giá trị số



- Tiếp theo dữ liệu tại cột duration sẽ được chia khoảng dựa trên khoảng tứ phân vị bọn em đã tính ở trên

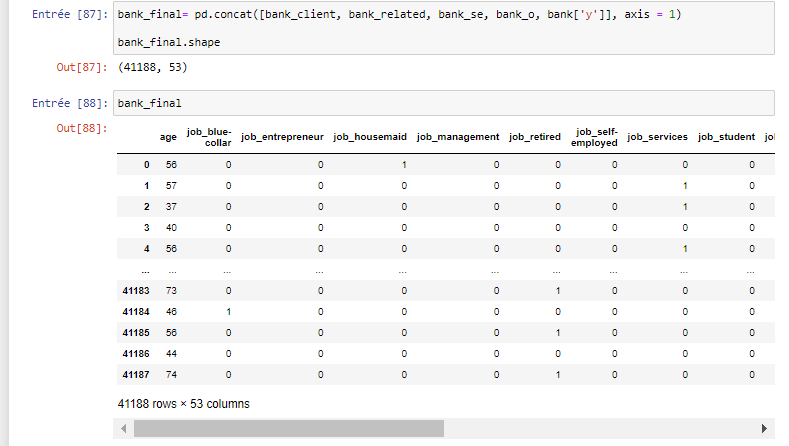


- Tiếp theo bọn em tiến hành tách nốt các cột còn lại thành các phần tương ứng



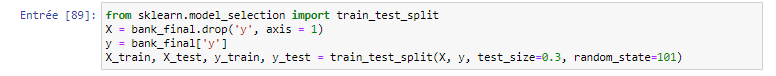
- Ở đây tất cả các cột còn lại đều là số trừ cột poutcome nên bọn em chỉ biến đổi mỗi cột đó thành số còn các cột khác dữ nguyên

- Cuối cùng bọn em ghép tất cả cột lại thành bảng hoàn chỉnh

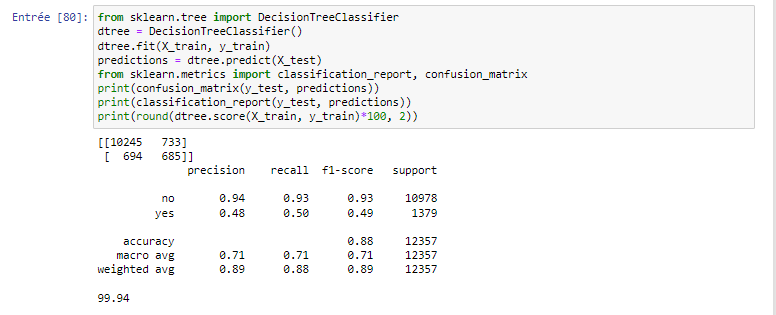


- Bảng cuối cùng sẽ bao gồm 41188 dòng và 53 cột

- Với dữ liệu này thì bọn em sẽ không có file train và test mà chỉ có 1 file nên chúng em sẽ dùng hàm train\_split\_test để tách random data dùng để train và test

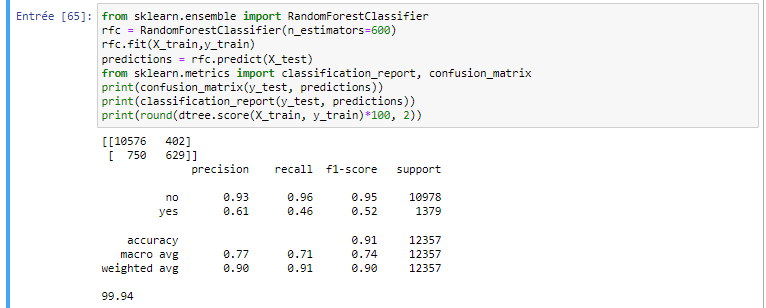


- Đầu tiên bọn em sẽ sử dùng mô hình Decision Tree



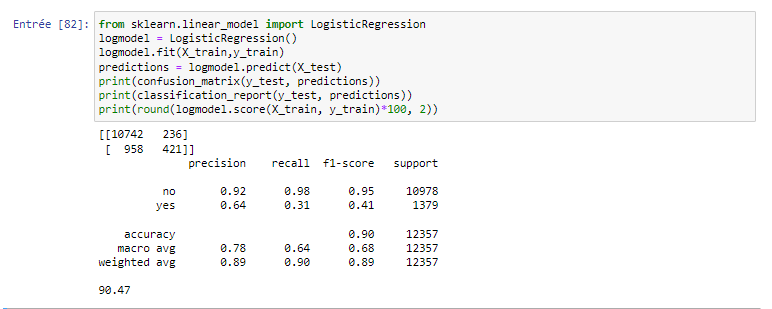
- Độ chính xác nếu dùng phương pháp này sẽ là 88%

- Tiếp theo là phương pháp Random Forest Trees với 600 cây random



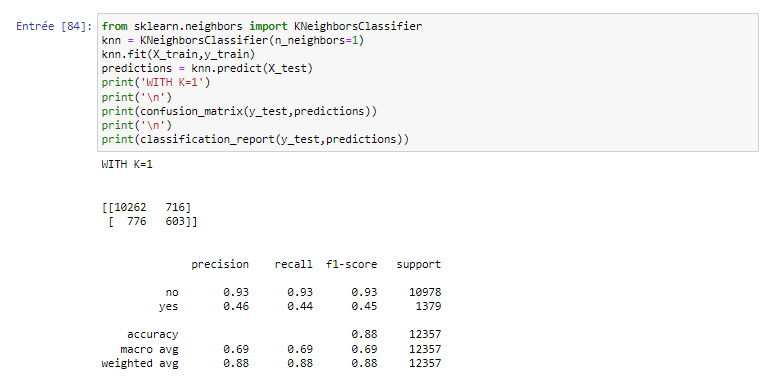
- Độ chính xác đã lên được 91%

- Tiếp theo là phương pháp LogisticRegression

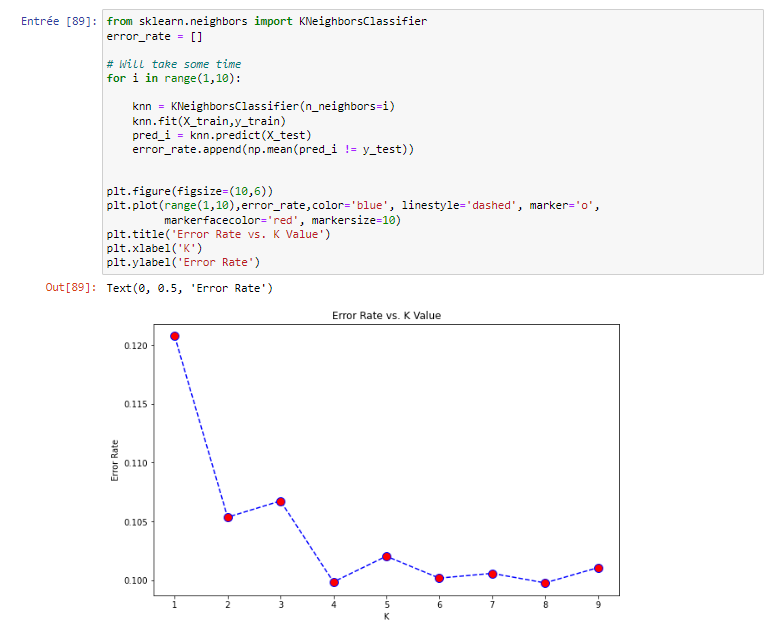


- Độ chính xác ở đây cũng lên tới 90%

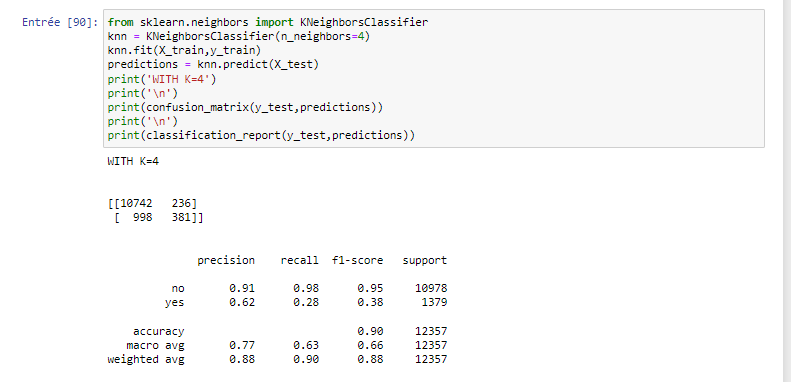
- Tiếp theo là phương pháp KNN với K=1



- Tiếp theo ta sẽ chạy từ 1 đến 10 điểm tìm giá trị K đưa đến error rate nhỏ nhất

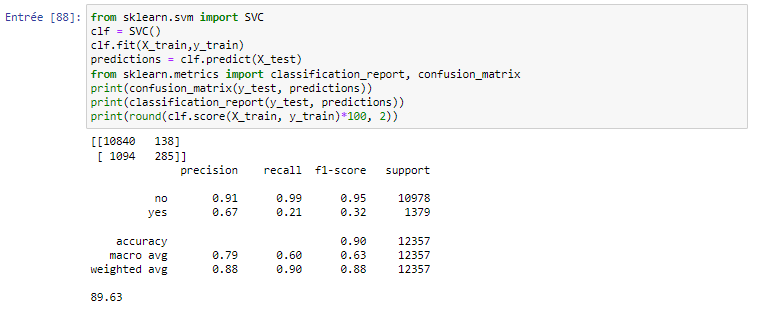


- Ta nhận thấy Error Rate nhỏ nhất ở K=4 nên ta chạy lại mô hình KNN với K=4



- Ta thấy độ chính xác đã lên được 90%!

- Tiếp theo là phương pháp SVC



- Độ chính xác ở đây là 90%

=> Có thể thấy với data này thì gần như các phương pháp đều cho ta độ chính xác như nhau nhưng vì Decision tree sẽ chạy nhanh hơn nên với những dataset loại này ta nên dùng phương pháp Decision tree và Random Forrest.