1. **Thu thập dữ liệu**
2. **Các thư viện và phần mềm cần cài đặt**

* Tải Chrome Driver tương ứng với phiên bản trên máy tính tại trang web: <https://chromedriver.chromium.org/downloads>
* Tải thư viện Selenium và Beautiful Soup 4:
  + ! pip install selenium
  + ! pip install bs4

1. **Chạy chương trình**

* Thu thập đường dẫn của dữ liệu:
  + Sử dụng Selenium để xử lí các trường hợp kéo xuống và nút nhấn See more để tải thêm dữ liệu trên trang web
  + Do việc thu thập dựa vào 10 loại Shape của kim cương nên cần truyền tham số đầu vào cho hàm Collect\_Href\_of\_Diamonds(shape, num) gồm giá trị Shape tương ứng và số lượng mẫu cần thu thập. Tuy nhiên, đôi lúc do việc khởi động trang web trong quá trình thu thập thì số lượng có thể ít hơn hoặc nhiều hơn dữ liệu cần thu thập
  + Sau khi thu thập được đường dẫn, ghi vào file .txt tương ứng với Shape
* Thu thập dữ liệu:
  + Sau khi có được các đường dẫn, tiến hành truy cập vào đường dẫn và lấy thông tin dữ liệu bằng BeautifulSoup thông qua hàm Collect\_Raw\_Data(url)
* Sau 2 bước thu thập trên, thu được dữ liệu thô (Raw\_Data) dưới dạng file .txt. Sử dụng hàm Convert line to CSV(line, columns) với tham số là dòng dữ liệu trong file và danh sách các cột dữ liệu để chuyển đổi dưới định dạng file .csv

**II. Xử lý và phân tích dữ liệu**

Chạy file ***3\_1\_Visualize\_Analyst\_(small).ipynb cho small\_data***, file ***3\_2\_Visualize\_Analyst\_(big).ipynb cho big\_data*** theo các bước sau:

* Chạy cell Import & function để import các thư viện và hàm cần thiết
* Chạy cell Overview để load data , xem các thông tin chung
* Chạy cell EDA để thực hiện xử lí data, phân tích data, tương quan và các mối quan hệ giữa các biến
* Chạy cell Feature selection để lựa chọn đặc trưng
* Chạy cell Label Encoder để chuyển các biến phân loại thành dạng số
* Chạy cell Feature transform để transform các biến

**III. Mô hình dự đoán**

Chạy file ***4.Model.ipynb*** để thực hiện mô hình dự đoán giá kim cương

1. Load dữ liệu từ 2 file: ***data\_small\_clean.csv*** (dành cho data 1000 samples) hoặc ***data\_big\_clean.csv*** (dành cho data 10000 samples).
2. Import các thư viện cần thiết
3. Chạy các cell code theo từng cụm đã chia sẵn từ trên xuống dưới.

* **Chú ý:**
* Ở những cell có ***“final\_result”*** thì bỏ chú thích và run cell. Chú ý, chỉ chạy 1 lần để tránh bị nhầm lẫn kết quả. Ví dụ:

# temp\_lr\_df = ['10.000', 'Random Forest Regression (RandomizedSearchCV)', run\_time\_big[1], improved\_model\_r2\_compare\_big.iloc[2, 1]\*100]

# final\_result.append(temp\_lr\_df)

final\_result

* Tương tự với cell mô hình trong phần  ***“Hyperparamter Tuning using RandomizedSearchCV và GridSearchCV”***, chỉ chạy 1 lần với mỗi cell. Nếu muốn không bị ghi quá nhiều kết quả dự đoán vào List kết quả thì hãy chạy lại những cell này trên mỗi tập dữ liệu tương ứng:

from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV

rf\_base\_r2\_big = max(result\_r2\_compare\_df\_big.sort\_values(by='R2 Testing Score', ascending=False)['R2 Testing Score'])

# List of improvement R2 Score using Grid Search with Cross Validation

improved\_test\_r2\_score\_big = [rf\_base\_r2\_big]

# List to track the run time of improvement models

run\_time\_big = []

* Khi chạy cell ở ***“Hyperparamter Tuning using RandomizedSearchCV và GridSearchCV”*** sẽ tốn khá nhiều thời gian thực thi. Nếu muốn thì có thể dùng sẵn Hyperparameter đã thử nghiệm để tiết kiệm thời gian.