# Auto Scout Project

# Project Description

## Objective:

The goal of this project is to clean and prepare an online vehicle dataset for machine learning modeling. The dataset consists of 15,919 rows and 54 columns from the year 2019, including various details about multiple car models (9 models), most of which are in inconsistent or broken formats. The main objective is to fix inconsistencies, handle missing values, and convert the raw data into a clean, structured, and numerically usable format.

The workflow involves four main preprocessing stages: data cleaning, missing value analysis, outlier detection, and dummy variable generation. For each stage, a separate notebook will be created. At the end of each notebook, the cleaned data will be saved as a `.csv` file. For example, after the **Data Cleaning** stage, a file named `clean_data.csv` will be generated. This file will be loaded at the beginning of the **Missing Value Analysis** notebook.

In the final step, a fully prepared dataset will be created to be used for **Machine Learning modeling**. All columns in this final dataset will be of **float** or **integer** types, ensuring compatibility with ML algorithms. The final DataFrame will contain approximately **15,000+ rows** and around **100 columns**.

## Project Phases:

### 1. Data Cleaning:

This phase involves removing broken, irrelevant, or redundant columns, and generating new columns with meaningful values.

Examples:

- Removing strings such as “km” and extracting pure numeric values.
- Splitting complex list values in the `Type` column to form new columns like `Fuel` and `Gears`.
- Removing columns with too many null values like `kW`.

- Creating new columns like `Paint_Type`, `Upholstery_type`, `Nr_of_Doors`, etc., through string parsing or external lookup.

Some columns appear duplicated or provide overlapping information under slightly different names. In such cases, these columns should be **merged** by combining their useful data into a single unified column. After merging, the redundant or less informative version of the column should be **dropped** to avoid duplication and confusion in later stages.

For example:

- The `prev_owner` and `previous_owners` columns should be consolidated into a single `previous_owners` column.

- Columns with similar or overlapping content should be carefully compared and cleaned using domain knowledge.

**Note:** Columns like `Comfort_Convenience`, `Entertainment_Media`, `Extras`, and `Safety_Security` are not altered during this step, as they will be transformed later using `get_dummies()`.

**2. Missing Value Analysis:**

Missing values are handled using several techniques. For example:

**Mode Imputation by Group:**

```
for group in df["make_model"].unique():

    cond = df["make_model"] == group

    mode_val = df[cond]["body_type"].mode()

    if not mode_val.empty:

        df.loc[cond, "body_type"] = df.loc[cond, "body_type"].fillna(mode_val[0])
```

**Multi-Level Group Filling:**

Grouping by make_model, age, and others to fill values in context.

## 3. Outlier Analysis

Outliers are detected and visualized using **boxplots** and **histograms**. In this project, the emphasis is not on removing all outliers, as some high or low values may be valid due to the nature of certain car models (e.g., premium vehicles with higher prices or powerful engines).

Instead of dropping these outliers, the goal is to identify **potential misprints or logically inconsistent values**. These suspicious values (e.g., hp = 0, displacement = 16000, weight = 0) are **converted to `np.nan`**, treating them as missing data.

Once these values are marked as `NaN`, we apply the same imputation techniques used in the **Missing Value Analysis** phase.

This approach preserves potentially meaningful outliers while handling unrealistic data responsibly.

**4. Dummy Variable Analysis:**

Categorical variables are transformed into numeric using `get_dummies()`:

```
pd.get_dummies(df, columns = [...])
```

```
df["Extras"].str.get_dummies(sep=",")
```

This is especially effective for multi-label fields like `Comfort_Convenience` and `Extras`.

## Key Functions and Techniques:

- **`df.apply()` with `lambda`**: Custom column transformations.
- **`re.extract()` / `re.sub()/re.findall()`**: Regex functions to extract numbers from strings.
- **`groupby()` with `fillna()`**: Imputation based on grouped logic.
- **`str.get_dummies()`**: Converts string-based list columns into binary indicator variables.
- **Visualization**:
    - `sns.boxplot()` — Outlier detection
    - `sns.heatmap()` — Correlation analysis

## Examples for key functions:

```
df['displacement'].str[0].str.replace(',','').str.findall('\d+').str[0].astype('float')
```

```
def inspection(a):
```

```
    if type(a) == list:
        return a[0].replace('\n', '')
    elif type(a) == str:
        return a.replace('\n', '')
    else:
        return a
```

### Columns:

**General Columns**

* url: url of dfs

* short_description, description: Description of dfs (in English and German) written by users

**Categorical Columns**

* make_model, make, model: Model of dfs. Ex:Audi A1

* body_type, body: Body type of dfs Example: van, sedans

* vat: VAT deductible, price negotiable

* registration, first_registration: First registration date and year of dfs.

* prev_owner, previous_owners: Number of previous owners

* type: new or used

* next_inspection, inspection_new: information about inspection (inspection date,..)

* body_color, body_color_original: Color of df Ex: Black, red

* paint_type: Paint type of df Ex: Metallic, Uni/basic

* upholstery: Upholstery information (texture, color)

* gearing_type: Type of gear Ex: dfmatic, manual

* fuel : fuel type Ex: diesel, benzine

* co2_emission, emission_class, emission_label: emission information

* drive_chain: drive chain Ex: front,rear, 4WD

* consumption: consumption of df in city, country and combination (lt/100 km)

* country_version

* entertainment_media

* safety_security

* comfort_convenience

* extras

**Quantitative Columns**

* price: Price of cars

* km: km of dfs

* hp: horsepower of dfs (kW)

* displacement: displacement of dfs (cc)

* warranty: warranty period (month)

* weight: weight of df (kg)

* nr_of_doors: number of doors

* nr_of_seats : number of seats

* cylinders: number of cylinders

* gears: number of gears