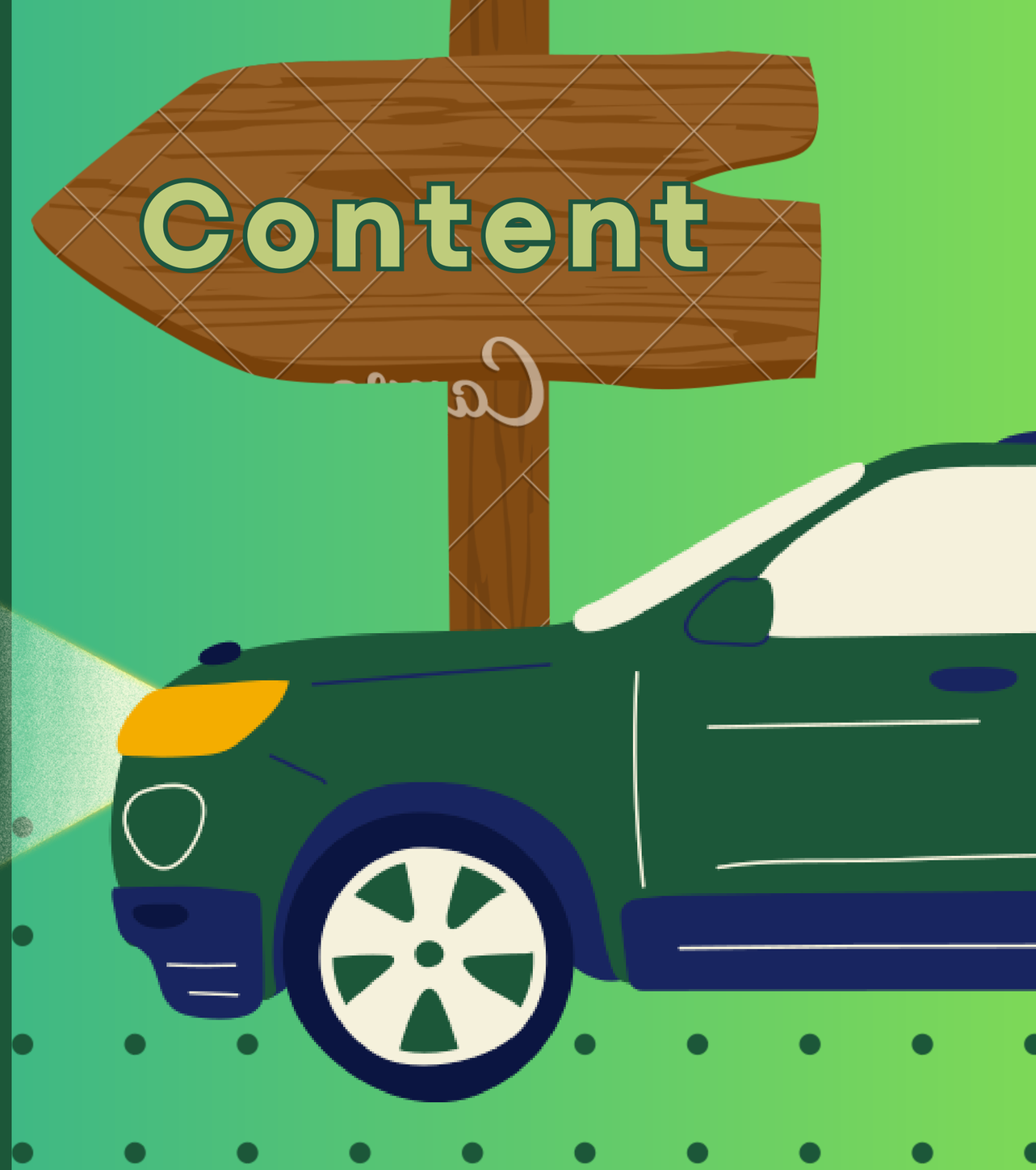
CAR PRICE PREDICT

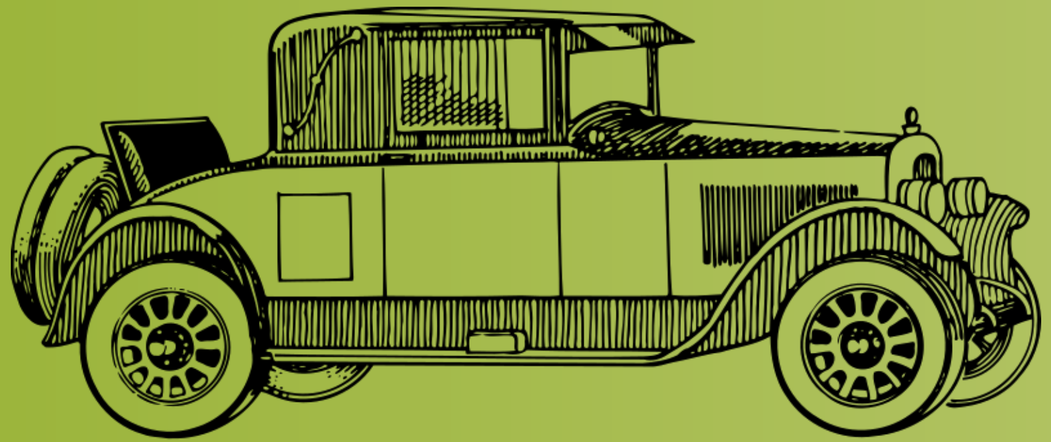
Nguyễn Hiền Nhân
DA20

Mentor: Hiền Đào

Content



DATA DESCRIPTION



Data description

- Bộ dữ liệu này chứa một số thông tin về ô tô đã qua sử dụng được liệt kê trên www.cardekho.com
- Bao gồm 8128 dòng và 13 cột

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.0
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5.0
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5.0
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0

8128 rows × 13 columns

Data description

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           8128 non-null   object 
 1   year           8128 non-null   int64  
 2   selling_price  8128 non-null   int64  
 3   km_driven      8128 non-null   int64  
 4   fuel           8128 non-null   object 
 5   seller_type    8128 non-null   object 
 6   transmission   8128 non-null   object 
 7   owner          8128 non-null   object 
 8   mileage        7907 non-null   object 
 9   engine         7907 non-null   object 
 10  max_power      7913 non-null   object 
 11  torque         7906 non-null   object 
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

Column

1. name: Tên của xe
2. year: Năm mà chiếc xe được mua
3. selling_price: Giá của chiếc xe đang được bán
4. km_driven: Số kilomet chiếc xe đã đi
5. fuel: Loại nhiên liệu của xe (xăng / diesel / CNG / LPG / điện)
6. seller_type: Cho biết Người bán là Cá nhân hay Đại lý
7. transmission: Cho biết hộp số của xe (Tự động / Thủ công)
8. owner: Số lượng chủ sở hữu trước đó
9. mileage: đơn vị tính số Kilomet đi được của xe đó với 1lit chất đốt
10. engine: Dung tích động cơ của xe tính bằng CC
11. max_power: Công suất tối đa của động cơ
12. torque: mô-men xoắn của xe
13. seats: Số chỗ ngồi trên xe

DATA PROCESSING

Data processing

name

có thể thấy các hãng xe đều có chữ đầu tiên trong tên là Brand của hãng xe
(ví dụ : Honda City 2017-2020 EXi thì Brand là Honda)

```
df['name']
```

```
0              Maruti Swift Dzire VDI
1       Skoda Rapid 1.5 TDI Ambition
2           Honda City 2017-2020 EXi
3          Hyundai i20 Sportz Diesel
4              Maruti Swift VXI BSIII
                     ...
8123                Hyundai i20 Magna
8124           Hyundai Verna CRDi SX
8125           Maruti Swift Dzire ZDi
8126                 Tata Indigo CR4
8127                 Tata Indigo CR4
Name: name, Length: 8128, dtype: obje
```

```
def get_car_brand():
    car_brand_list= []
    for name in df['name']:
        car_brand_list.append(name.split(' ')[0])
    return car_brand_list


df['brand'] = get_car_brand()
```

```
df['brand']
```

```
0         Maruti
1          Skoda
2          Honda
3        Hyundai
4         Maruti
         ...
8123     Hyundai
8124     Hyundai
8125      Maruti
8126        Tata
8127        Tata
Name: brand, Length: 8128,
```

Data processing

year

```
df['year']
```

```
0       2014
1       2014
2       2006
3       2010
4       2007
        ...
8123    2013
8124    2007
8125    2009
8126    2013
8127    2013
Name: year, Leng
```

có thể tính tuổi của xe bằng cách lấy năm hiện tại trừ năm sản xuất

```
df["car_age"] = (datetime.datetime.now().year) - (df["year"])
```

```
df["car_age"]
```

```
0        9
1        9
2       17
3       13
4       16
        ..
8123    10
8124    16
8125    14
8126    10
8127    10
Name: car_age, L
```

mileage

```
df['mileage']
```

```
0        23.4 kmpl
1       21.14 kmpl
2        17.7 kmpl
3        23.0 kmpl
4        16.1 kmpl
           ...
8123     18.5 kmpl
8124     16.8 kmpl
8125     19.3 kmpl
8126    23.57 kmpl
8127    23.57 kmpl
Name: mileage, Leng
```

chuyển từ dạng object có đơn vị 'kmpl' về dạng số
thay thế giá trị null bằng giá trị mean

```
df["mileage"] = df["mileage"].str.extract('([^\s]+)').astype("float")

df["mileage"].replace(np.nan, "%.3f" % df["mileage"].astype("float").mean(axis=0), inplace=True)

df["mileage"] = df["mileage"].astype("float")
```

```
df["mileage"]
```

```
0       23.40
1       21.14
2       17.70
3       23.00
4       16.10
        ...
8123    18.50
8124    16.80
8125    19.30
8126    23.57
8127    23.57
Name: mileage, Le
```

Data processing

Làm tương tự với column 'max_power' và 'engine' ta có

```
df["max_power"]
```

```
0        74.00
1       103.52
2        78.00
3        90.00
4        88.20
        ...
8123     82.85
8124    110.00
8125     73.90
8126     70.00
8127     70.00
Name: max_power, Length: 8128, dtype: float64
```

```
df["engine"]
```

```
0       1248.0
1       1498.0
2       1497.0
3       1396.0
4       1298.0
        ...
8123    1197.0
8124    1493.0
8125    1248.0
8126    1396.0
8127    1396.0
Name: engine, Length: 8128, dtype: float64
```

Với column 'seats' ta thay thế giá trị null bằng giá trị xuất hiện nhiều nhất

```
df["seats"].replace(np.nan, df['seats'].value_counts().idxmax(), inplace=True)
df['seats'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 8128 entries, 0 to 8127
Series name: seats
Non-Null Count  Dtype
--------------  -----
8128 non-null   float64
dtypes: float64(1)
memory usage: 63.6 KB
```

Data processing

Drop các column không cần thiết và check null

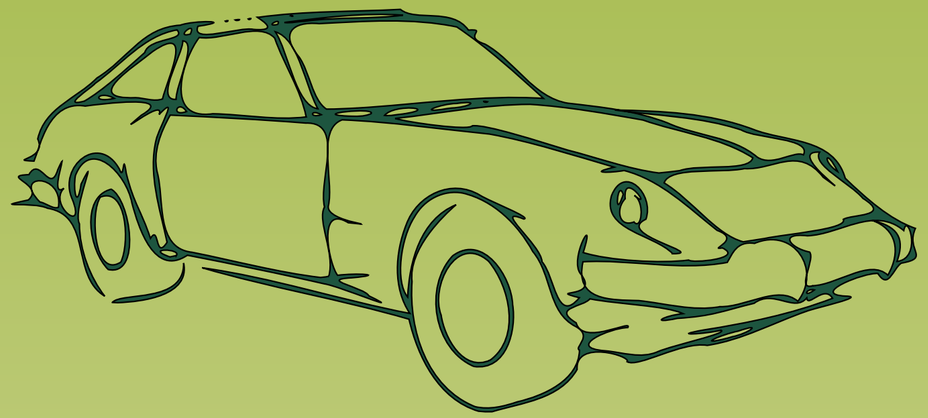
```
df.drop(['torque'], axis = 1, inplace = True)
df.drop(['year'], axis = 1, inplace = True)
df.drop(['name'], axis = 1, inplace = True)
```

```
df.info()
```

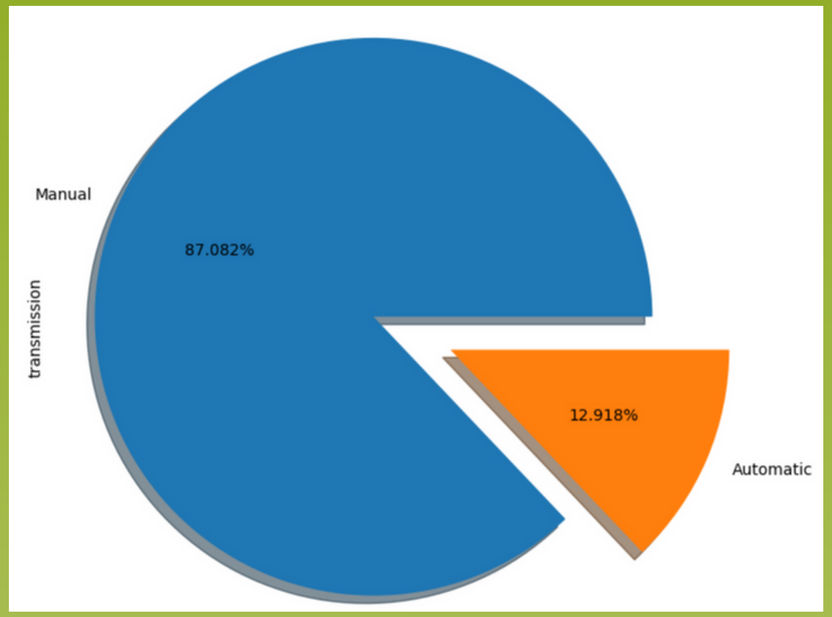
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   selling_price  8128 non-null   int64
 1   km_driven      8128 non-null   int64
 2   fuel           8128 non-null   object
 3   seller_type    8128 non-null   object
 4   transmission   8128 non-null   object
 5   owner          8128 non-null   object
 6   mileage        8128 non-null   float64
 7   engine         8128 non-null   float64
 8   max_power      8128 non-null   float64
 9   seats          8128 non-null   float64
 10  brand          8128 non-null   object
 11  car_age        8128 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 762.1+ KB
```

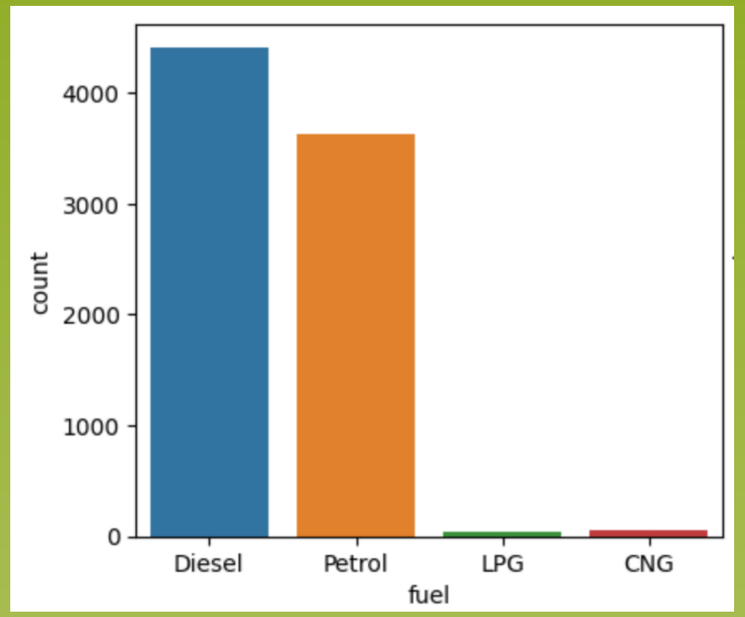
Dữ liệu đã khá sạch, cùng sang bước **EDA**

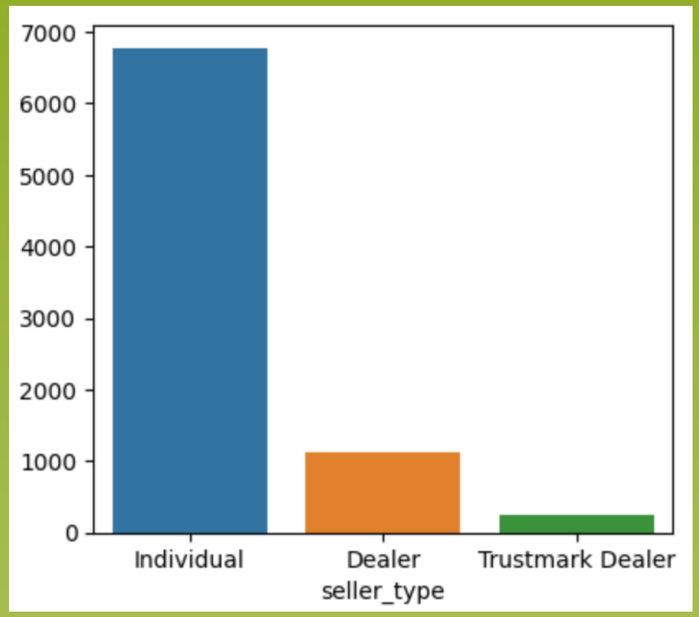
EXPLORATORY DATA ANALYSIS



EDA

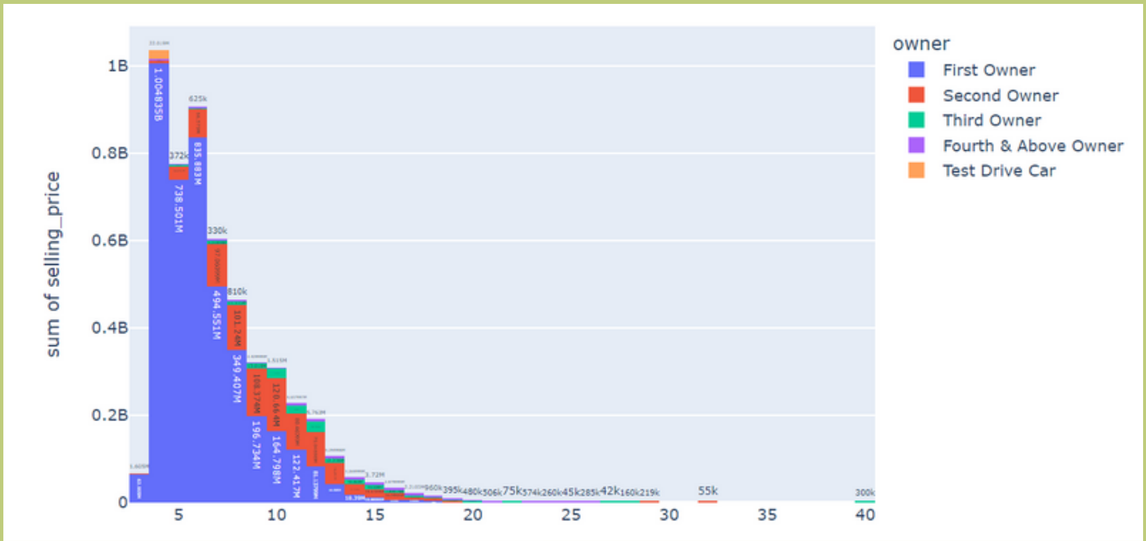


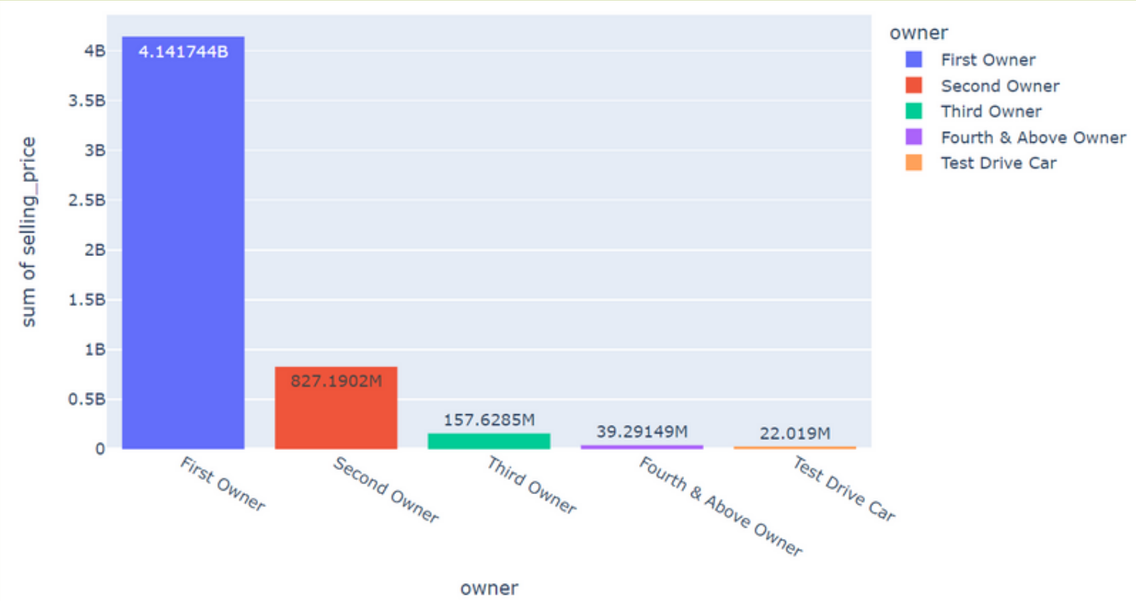




- Khách đa phần yêu thích loại xe số sàn (87%) so với số tự động (12.9%)
- 2 loại nguyên liệu chính được phổ biến trên các dòng xe là Diesel và Petrol
- Khách hàng mua xe qua đại lý nhiều hơn so với việc mua xe qua cá nhân

EDA

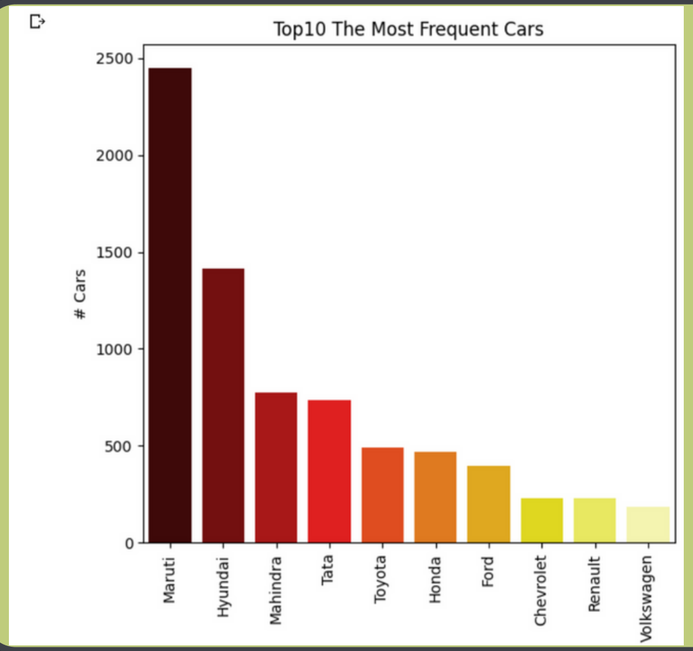




- Xe được mua nhiều nhất từ năm thứ 4 trở đi ,
- Doanh thu cao nhất đến từ xe đã 1 người sử dụng.

➡ Tiêu chí chọn xe của khách hàng khi chọn xe cũ là xe khoảng 4 ,5, 6 năm có thể lúc này là lúc mà giá chiếc xe thích hợp nhất so với giá trị của chiếc xe , hơn nữa mới chỉ qua tay của 1 đến 2 người thì chiếc xe không bị xuống cấp quá nhiều

EDA

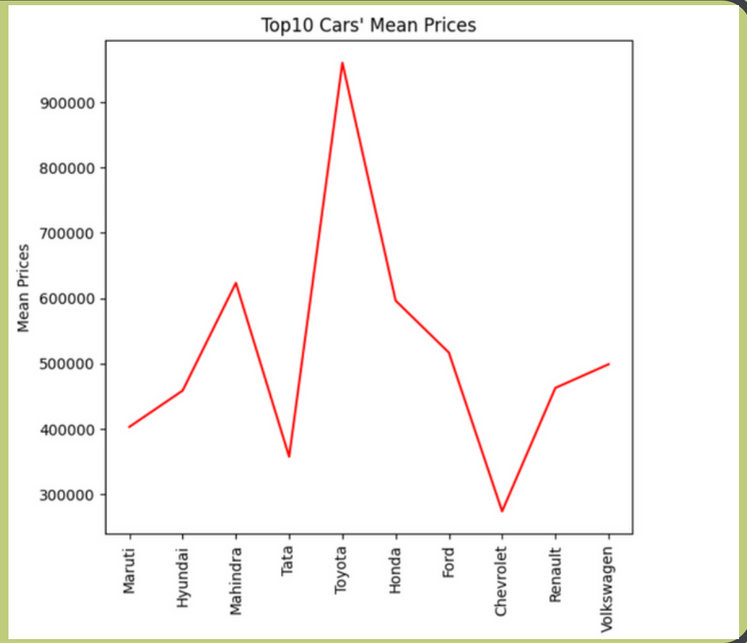


Top 10 hãng xe bán chạy nhất

Maruti	2.448
Hyundai	1.415
Mahindra	772
Tata	734
Toyota	488
Honda	467
Ford	397
Chevrolet	230
Renault	228
Volkswagen	186

Giá xe trung bình của 10 hãng xe trên

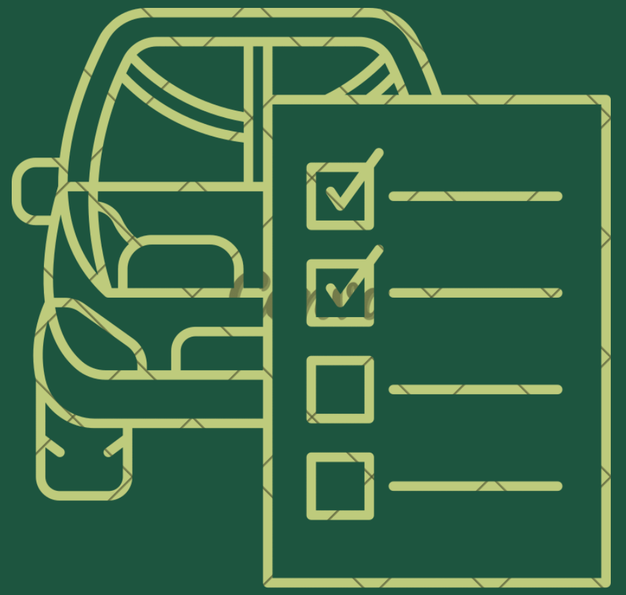
Maruti	403.075
Hyundai	458.554
Mahindra	623.224
Tata	357.433
Toyota	959.946
Honda	596.178
Ford	516.682
Chevrolet	273.867
Renault	462.618
Volkswagen	498.817



- Hãng xe bán chạy nhất là Maruti và bỏ xa so với các hãng xe còn lại
- Hãng xe Maruti này lại có giá xe trung bình lại không cao

→ Khách hàng chọn xe vì nó có giá cả hợp lý , không quá xa xỉ , chất lượng ổn định

DATA PREPARATION & TRAINING MODEL



Data preparation & Training model

- giữa các biến 'mileage' và 'engine' : 0,57
- giữa các biến 'max_power' và 'engine' : 0,70
- giữa các biến 'seats' và 'engine' : 0,61
- giữa các biến 'max_power' và 'selling_price' : 0,74
- Cũng có mức độ tương quan trung bình giữa các biến khác.

Data preparation & Training model

Chọn biến đầu vào X và biến target y

```
X = df.drop("selling_price", axis = 1)
y = df['selling_price']
```

LabelEncoder với các biến categorical

```
le = LabelEncoder()
categorical_feature_mask = df.dtypes==object
categorical_cols = df.columns[categorical_feature_mask].tolist()
df[categorical_cols] = df[categorical_cols].apply(lambda col: le.fit_transform(col))
```

StandardScaler X

```
X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))
```

Data preparation & Training model

Chia tập train và tập test

```
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.25,random_state=42)
print(Xtrain.shape,ytrain.shape,Xtest.shape,ytest.shape)
```

```
(6096, 11) (6096,) (2032, 11) (2032,)
```

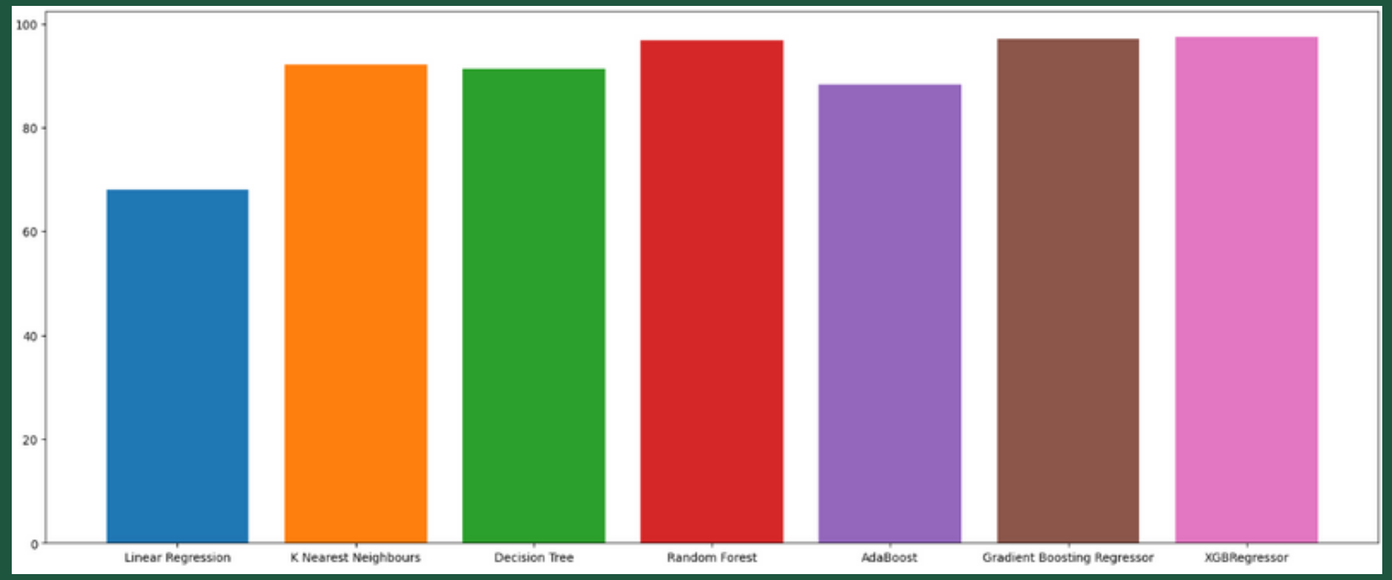
Chạy với các mô hình
với thông số như hình

```
lr  = LinearRegression()
knn = KNeighborsRegressor(n_neighbors=8)
dt  = DecisionTreeRegressor(max_depth = 5)
rf  = RandomForestRegressor(n_estimators=100, max_features= 7)
ada = AdaBoostRegressor( n_estimators=150, learning_rate =.08)
gbr = GradientBoostingRegressor(max_depth=7, n_estimators=500, learning_rate =.05)
xgb = XGBRegressor(max_depth = 7, n_estimators=500, learning_rate =.05)
```

Data preparation & Training model

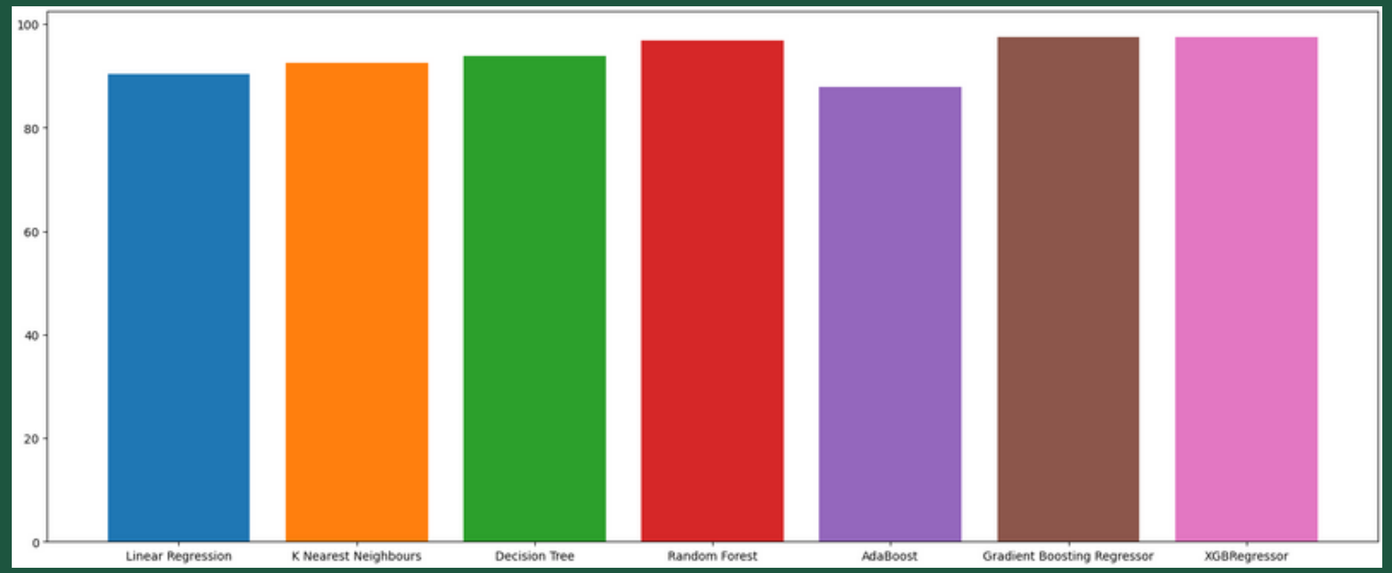
Kết quả

Linear Regression : 68 %
K Nearest Neighbours : 92 %
Decision Tree : 91 %
Random Forest : 97 %
AdaBoost : 88 %
Gradient Boosting Regressor : 97 %
XGBRegressor : 98 %



Dùng Polynomial (Degree= 2) ta được kết quả như sau

Linear Regression : 90 %
K Nearest Neighbours : 93 %
Decision Tree : 94 %
Random Forest : 97 %
AdaBoost : 88 %
Gradient Boosting Regressor : 98 %
XGBRegressor : 98 %




Thank You