ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

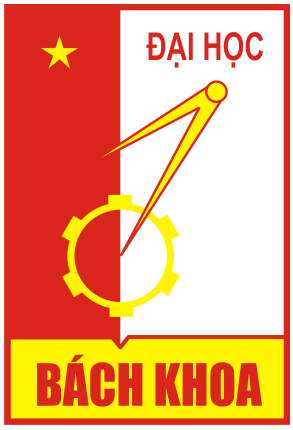
KHOA TOÁN – TIN
Faculty Of Applied Mathematics And Informatics

CAR PRICE

HỌC PHẦN: HỆ HỖ TRỢ QUYẾT ĐỊNH

Nhóm sinh viên thực hiện: Nhóm 25
Lê Ngọc Hà - 20216922
Nguyễn Thị Linh Chi - 20216913

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



CHỦ ĐỀ: CAR PRICE

HỌC PHẦN: HỆ HỖ TRỢ QUYẾT ĐỊNH

Giảng viên hướng dẫn: TS. Lê Hải Hà

Nhóm sinh viên thực hiện: Nhóm 25 - 150330

Lê Ngọc Hà 20216922
Nguyễn Thị Linh Chi 20216913

HÀ NỘI, 06/2024

Lời mở đầu

Lời đầu tiên, nhóm báo cáo xin gửi lời cảm ơn chân thành tới thầy TS. Lê Hải Hà, người đã trực tiếp giảng dạy và hướng dẫn chúng em trong học phần "Hệ hỗ trợ quyết định" . Từ những kiến thức trên lớp học hỏi được, nhóm đã vận dụng chúng vào trong bài báo cáo này.

Trong thời đại công nghệ 4.0, việc ứng dụng các phương pháp và công cụ hiện đại để tối ưu hóa quy trình kinh doanh và cung cấp các dịch vụ tiện ích cho người tiêu dùng trở nên vô cùng cần thiết. Một trong những lĩnh vực đang chứng kiến sự thay đổi mạnh mẽ nhờ vào sự phát triển của trí tuệ nhân tạo (AI) và học máy (Machine Learning) chính là thị trường ô tô, đặc biệt là trong việc dự đoán giá xe.

Thị trường ô tô với tính phức tạp và biến động không ngừng, đòi hỏi các công cụ phân tích và dự đoán chính xác để giúp cả người mua và người bán đưa ra những quyết định sáng suốt. Giá trị của một chiếc xe không chỉ phụ thuộc vào các yếu tố cơ bản như năm sản xuất, hãng sản xuất, dòng xe mà còn bị ảnh hưởng bởi nhiều yếu tố khác như tình trạng xe, số km đã đi, tình hình thị trường và xu hướng tiêu dùng.

Trong bối cảnh đó, bài toán dự đoán giá xe sử dụng các thuật toán học máy không chỉ giúp xác định giá trị thực của xe dựa trên dữ liệu lịch sử mà còn cung cấp cái nhìn tổng quan về các yếu tố ảnh hưởng chính đến giá cả. Việc áp dụng các thuật toán này có thể giúp nâng cao hiệu quả kinh doanh cho các đại lý ô tô, trang web thương mại điện tử và cả các công ty bảo hiểm xe.

Bài báo cáo không tránh khỏi những sai sót do kiến thức còn hạn hẹp, chúng em rất mong nhận được ý kiến đánh giá và góp ý từ thầy để bài làm được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

Hà Nội, ngày 23 tháng 6 năm 2024

Nhóm sinh viên thực hiện

NHÓM 25

Mục lục

1 Phát biểu bài toán

1.1 Mô tả bài toán

Xây dựng một hệ thống dự đoán giá xe dựa trên các đặc điểm kỹ thuật, tình trạng và các yếu tố liên quan khác nhằm cung cấp một công cụ mạnh mẽ và chính xác cho người tiêu dùng, người bán và các đại lý xe hơi. Mục tiêu chính của hệ thống này là tạo ra một mô hình học máy có thể phân tích các thuộc tính của xe và dự đoán giá trị thị trường hiện tại của xe một cách chính xác và đáng tin cậy. Hệ thống này sẽ giúp người dùng dễ dàng đánh giá giá trị của xe trong các giao dịch mua bán, từ đó tối ưu hóa quyết định mua bán xe.

1.2 Đầu vào của bài toán

Đầu vào của bài toán là một tập dữ liệu chứa thông tin chi tiết về các xe đã qua sử dụng hoặc xe mới bao gồm các thuộc tính sau:

1. Model.
2. Year.
3. Price.
4. Transmission.
5. Mileage.
6. FuelType.
7. Tax.
8. MPG.
9. EngineSize.
10. Manufacturer.

- Định dạng tệp tin: .csv
- Số lượng bản ghi: 97712
- Kích thước bộ dữ liệu: 5.44MB
- Số thuộc tính: 10

1.3 Đầu ra của bài toán

Đầu ra của bài toán là giá dự đoán của xe dựa trên các đặc điểm đầu vào.

1.4 Yêu cầu xử lý

Thu thập dữ liệu: Thu thập một bộ dữ liệu xe trên trang web Kaggle.

Thống kê dữ liệu mẫu: Sử dụng ngôn ngữ lập trình python để thống kê và trực quan hóa dữ liệu, từ đó giúp hiểu rõ về mối quan hệ giữa các biến độc lập và biến mục tiêu.

Xử lý và làm sạch dữ liệu:

- Loại bỏ khoảng trắng dư thừa trong cột Model.
- Chuyển cột Price từ kiểu int sang kiểu float.
- Biến đổi phân bố của cột Mileage.
- Kiểm tra và xử lý dữ liệu ngoại lệ của các cột sau: Mileage, Mpg, EngineSize.
- Chuyển đổi các dữ liệu định danh thành dạng số đối với các cột sau: Model, Transmission, FuelType và Manufacturer.
- Chuẩn hoá dữ liệu của tập dữ liệu huấn luyện và tập dữ liệu kiểm tra.

Chọn và xây dựng mô hình: Áp dụng thuật toán KNN để huấn luyện mô hình trên tập dữ liệu huấn luyện và điều chỉnh các tham số của mô hình.

Đánh giá mô hình: Sử dụng các chỉ số đánh giá có thể bao gồm: MSE (Mean Squared Error) và R^2 (R-squared).

Triển khai mô hình: Sau khi mô hình đạt được độ chính xác mong muốn, triển khai mô hình để dự đoán giá xe trên tập dữ liệu kiểm tra.

Ứng dụng mô hình: Mô tả ứng dụng mô hình (dự báo với dữ liệu mới hay mô tả các trường hợp ứng dụng khác của các mô hình không giám sát).

2 Tiền xử lý dữ liệu

2.1 Thu thập dữ liệu

Bộ dữ liệu "CarsData" được lấy từ trang web Kanggle, đây là một nền tảng trực tuyến dành cho các nhà khoa học dữ liệu, nhà phân tích dữ liệu khám phá, phân tích và chia sẻ dữ liệu. Bộ dữ liệu sử dụng được thu thập trong thời gian từ năm 1970 đến năm 2024, có định dạng file.csv gồm 97712 bản ghi và 10 thuộc tính.

model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize	Manufacturer
I10	2017	7495	Manual	11630	Petrol	145	60.1	1	hyundi
Polo	2017	10989	Manual	9200	Petrol	145	58.9	1	volkswagen
2 Series	2019	27990	Semi-Auto	1614	Diesel	145	49.6	2	BMW
Yeti Outdoor	2017	12495	Manual	30960	Diesel	150	62.8	2	skoda
Fiesta	2017	7999	Manual	19353	Petrol	125	54.3	1.2	ford
C-HR	2019	26791	Automatic	2373	Hybrid	135	74.3	1.8	toyota
Kuga	2019	17990	Manual	7038	Petrol	145	34.4	1.5	ford
Tiguan	2019	27490	Semi-Auto	3000	Petrol	145	30.4	2	volkswagen
Fiesta	2018	9891	Manual	31639	Petrol	145	65.7	1	ford
A Class	2017	17498	Manual	9663	Diesel	30	62.8	2.1	merc
Kuga	2017	16500	Semi-Auto	30000	Diesel	145	54.3	2	ford
1 Series	2016	10550	Manual	40313	Diesel	0	78.5	1.5	BMW
Up	2017	7990	Manual	9179	Petrol	145	64.2	1	volkswagen
Golf	2019	11980	Manual	14621	Petrol	145	49.6	1	volkswagen
Corsa	2017	7499	Manual	25000	Petrol	150	55.4	1.4	vauxhall
RAV4	2016	20790	Automatic	32196	Hybrid	20	55.4	2.5	toyota
Fiesta	2017	10490	Manual	16087	Petrol	0	65.7	1	ford
GLA Class	2016	15600	Automatic	42844	Diesel	30	64.2	2.1	merc
Golf	2017	17750	Manual	27125	Diesel	20	67.3	2	volkswagen
Aygo	2016	7130	Manual	23971	Petrol	0	69	1	toyota
Q5	2020	37000	Semi-Auto	1000	Diesel	145	38.2	2	Audi
Fiesta	2017	12750	Semi-Auto	12687	Petrol	145	54.3	1	ford
Karoq	2019	21490	Semi-Auto	4530	Diesel	145	47.1	1.6	skoda
Scala	2019	15490	Manual	5	Petrol	145	49.6	1	skoda
Auris	2017	11290	Manual	12890	Petrol	150	51.4	1.2	toyota
Corsa	2018	8499	Manual	7500	Petrol	145	55.4	1.4	vauxhall

Hình 2.1. Tổng quan về bộ dữ liệu "CarsData.csv"

Diễn giải thông tin về các trường dữ liệu:

- Model: Mẫu xe.
- Year: Năm sản xuất của ô tô.
- Price: Giá xe.
- Transmission: Loại hộp số được sử dụng trên ô tô.
- Mileage: Quãng đường đã đi được của ô tô.
- FuelType: Loại nhiên liệu mà ô tô sử dụng.
- Tax: Thuế suất áp dụng cho ô tô.
- MPG: Hiệu suất của ô tô trên mỗi gallon.
- EngineSize: Kích thước động cơ của ô tô.
- Manufacturer: Nhà sản xuất ô tô.

2.2 Thống kê dữ liệu mẫu

2.2.1 Import các thư viện

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import neighbors, datasets
import pandas as pd
from google.colab import drive
import seaborn as sns
drive.mount('/content/drive')
path = '/content/drive/MyDrive/CarsData.csv'
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from scipy.stats import boxcox
from sklearn.metrics import r2_score
```

2.2.2 Đọc dữ liệu từ file.csv

```
df = pd.read_csv(path)
print(df)
```

	model	year	price	transmission	mileage	fuelType	tax	mpg
0	I10	2017	7495	Manual	11630	Petrol	145	60.1
1	Polo	2017	10989	Manual	9200	Petrol	145	58.9
2	2 Series	2019	27990	Semi-Auto	1614	Diesel	145	49.6
3	Yeti Outdoor	2017	12495	Manual	30960	Diesel	150	62.8
4	Fiesta	2017	7999	Manual	19353	Petrol	125	54.3
...	...	...	...	...	...	...	...	...
97707	Fiesta	2017	10447	Automatic	8337	Petrol	145	54.3
97708	3 Series	2014	14995	Manual	25372	Diesel	30	61.4
97709	Fiesta	2017	8950	Manual	19910	Petrol	125	54.3
97710	Astra	2017	10700	Automatic	24468	Petrol	125	50.4
97711	Grandland X	2019	15798	Manual	10586	Diesel	150	48.7

```
       engineSize Manufacturer
0             1.0       hyundi
1             1.0   volkswagen
2             2.0          BMW
3             2.0        skoda
4             1.2         ford
...           ...          ...
97707         1.0         ford
97708         2.0          BMW
97709         1.2         ford
97710         1.4     vauxhall
97711         1.5     vauxhall

[97712 rows x 10 columns]
```

2.2.3 Khám phá dữ liệu

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97712 entries, 0 to 97711
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   model         97712 non-null  object
 1   year          97712 non-null  int64
 2   price         97712 non-null  int64
 3   transmission  97712 non-null  object
 4   mileage       97712 non-null  int64
 5   fuelType      97712 non-null  object
 6   tax           97712 non-null  int64
 7   mpg           97712 non-null  float64
 8   engineSize    97712 non-null  float64
 9   Manufacturer  97712 non-null  object
dtypes: float64(2), int64(4), object(4)
memory usage: 7.5+ MB
None
```

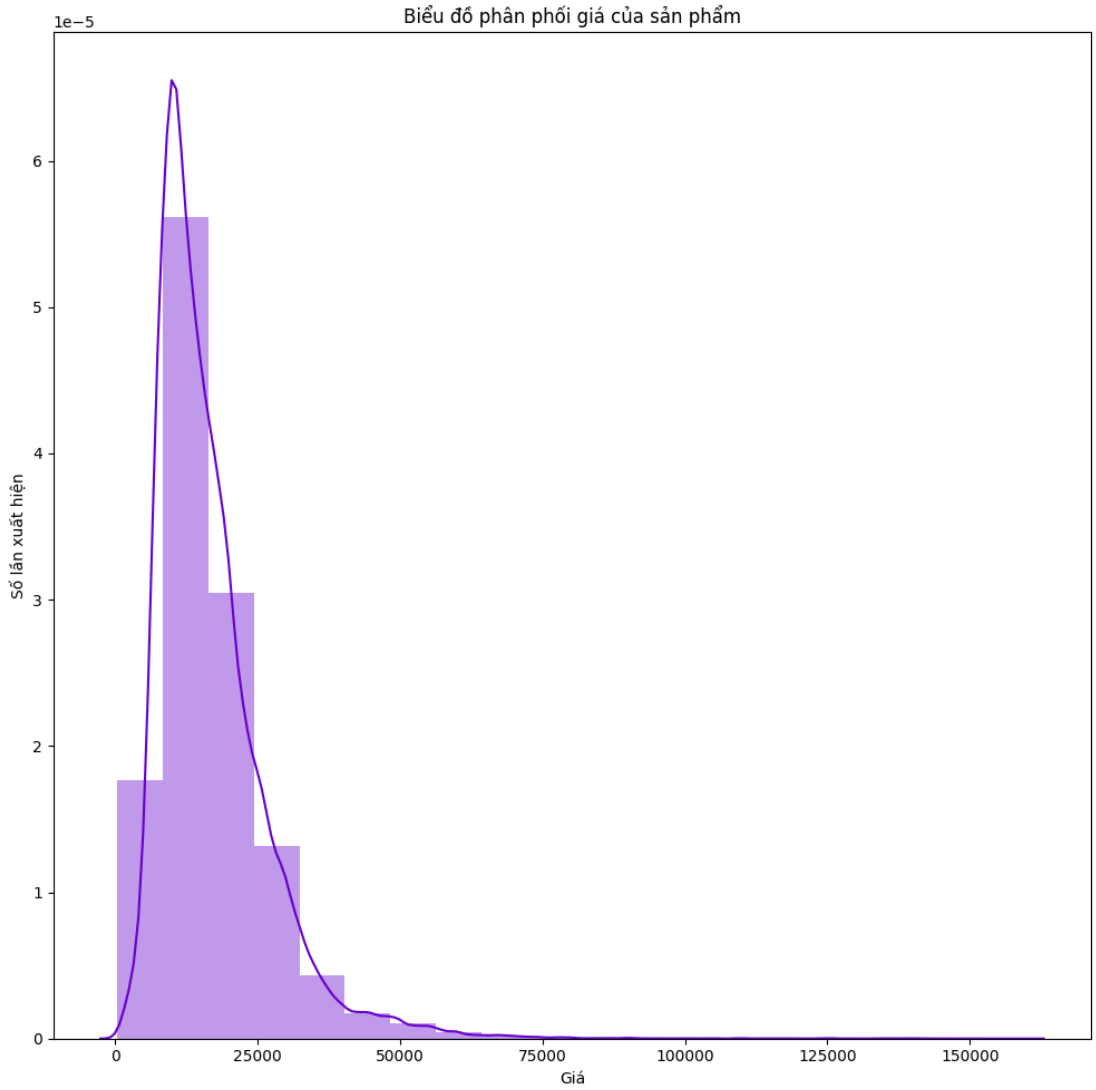
2.2.4 Thống kê mô tả cho các cột số học, phương sai, độ lệch chuẩn

```
    numeric_columns = df.select_dtypes(include=['int64', 'float64'
]).columns
    numeric_columns = numeric_columns.drop('year')
    df1 = df[numeric_columns]
    stats = df1.describe()
    variance = df1.var()
    std_dev = df1.std()
    stats.loc['variance'] = variance
    stats.loc['std_dev'] = std_dev
    print(stats)
```

	price	mileage	tax	mpg	engineSize
count	9.771200e+04	9.771200e+04	97712.000000	97712.000000	97712.000000
mean	1.677349e+04	2.321948e+04	120.142408	55.205623	1.664913
std	9.868552e+03	2.106088e+04	63.357250	16.181659	0.558574
min	4.500000e+02	1.000000e+00	0.000000	0.300000	0.000000
25%	9.999000e+03	7.673000e+03	125.000000	47.100000	1.200000
50%	1.447000e+04	1.768250e+04	145.000000	54.300000	1.600000
75%	2.075000e+04	3.250000e+04	145.000000	62.800000	2.000000
max	1.599990e+05	3.230000e+05	580.000000	470.800000	6.600000
variance	9.738832e+07	4.435608e+08	4014.141124	261.846094	0.312005
std_dev	9.868552e+03	2.106088e+04	63.357250	16.181659	0.558574

2.2.5 Biểu đồ phân phối giá của sản phẩm

```
plt.figure(figsize=(10,10))
rating= df.price.astype(float)
sns.distplot(rating, bins=20, color='#6600CC'),
plt.xlabel('Gia')
plt.ylabel('So lan xuat hien')
plt.title('Bieu do phan phoi gia cua san pham')
plt.grid(False)
plt.tight_layout()
plt.show()
```

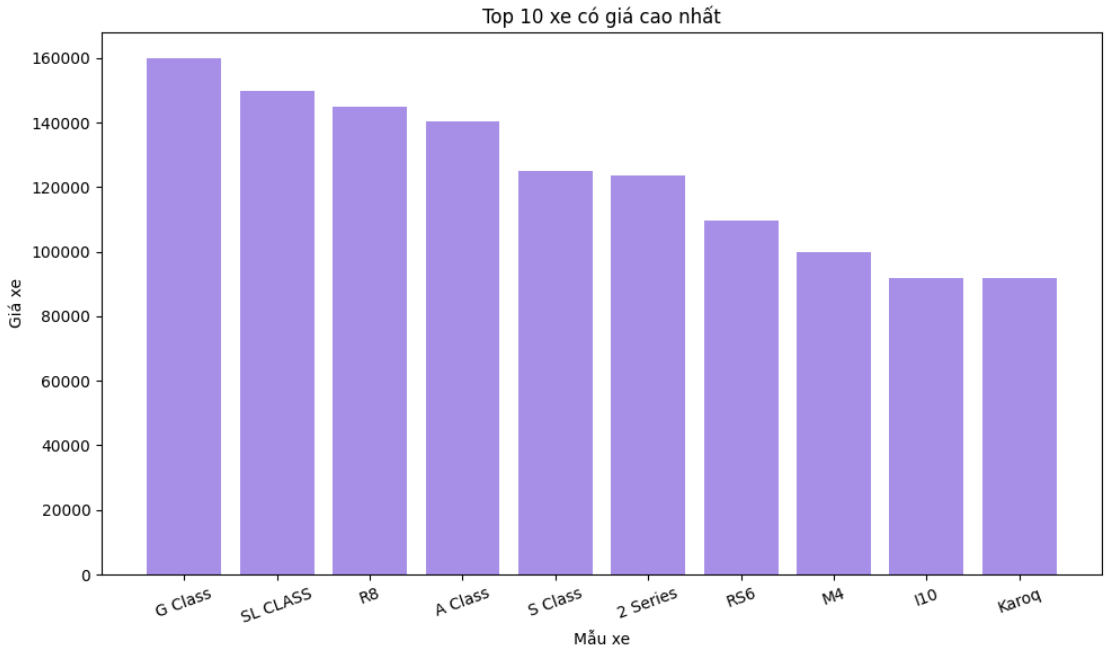


Hình 2.2. Biểu đồ phân phối giá sản phẩm

Nhận xét: Ta thấy giá của các loại xe chủ yếu rơi vào khoảng 10.000$ đến 40.000$.

2.2.6 Biểu đồ top 10 xe có giá cao nhất

```
top_10_expensive_cars = df.sort_values(by='price', ascending=
False).drop_duplicates(subset='model').head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_expensive_cars['model'], top_10_expensive_cars[
'price'], color='#A78FE7')
plt.xlabel('Mau xe')
plt.ylabel('Gia xe')
plt.title('Top 10 xe co gia cao nhat')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

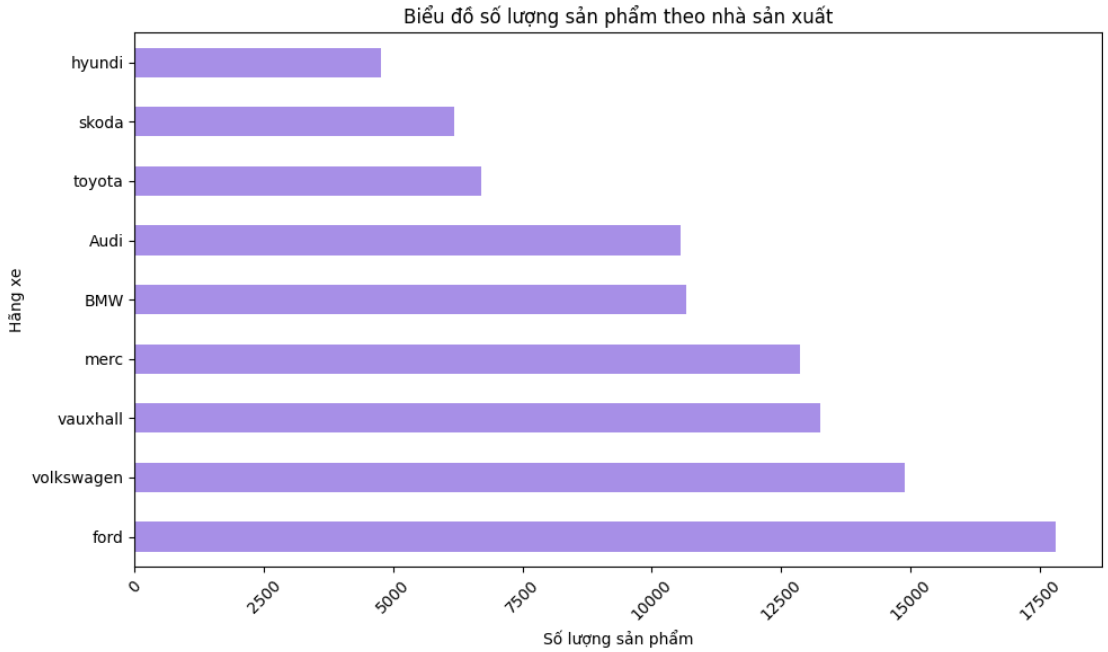


Hình 2.3. Biểu đồ top 10 xe có giá cao nhất

Nhận xét: Top 10 xe có giá cao nhất có thể kể đến như là G Class (160.000$), SL Class (khoảng 150.000$), R8 (khoảng 145.000$)... Ta nhận thấy có sự chênh lệch giá đáng kể giữa xe ở vị trí top 1 và top 10, chủ yếu các mẫu xe trong danh sách này thuộc phân khúc xe sang và xe thể thao cao cấp.

2.2.7 Biểu đồ số lượng sản phẩm theo nhà sản xuất

```
manufacturer_counts = df['Manufacturer'].value_counts()
plt.figure(figsize=(10, 6))
manufacturer_counts.plot(kind='barh', color='#A78FE7')
plt.ylabel('Hang xe')
plt.xlabel('So luong san pham')
plt.title('Bieu do so luong san pham theo nha san xuat')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Hình 2.4. Biểu đồ số lượng sản phẩm theo nhà sản xuất

Nhận xét: Các nhà sản xuất có số lượng xe nhiều nhất có thể kể đến như Ford, Volkswagen, Vauxhall, Merc... Đây đều là những hãng xe nổi tiếng nhất trên thị trường.

2.2.8 Biểu đồ scatter giữa Price và Tax

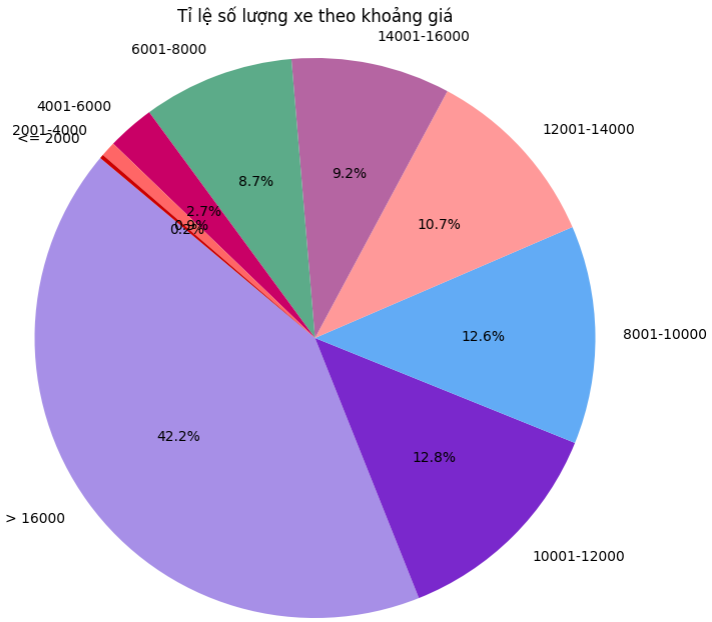
```
    plt.figure(figsize=(10, 6))
    plt.scatter(df['price'], df['tax'], alpha=0.5,c='#A78FE7',
   edgecolors='none')
    plt.title('Bieu do scatter giua Price va Tax')
    plt.xlabel('Price')
    plt.ylabel('Tax')
    plt.grid(False)
    plt.show()
```

Hình 2.5. Biểu đồ Scatter giữa Price và Tax

Nhận xét: Phần lớn dữ liệu tập trung ở mức giá dưới 60.000$. Có sự hiện diện của nhiều mức thuế cố định, tạo thành các dải ngang rõ rệt trên biểu đồ. Đáng chú ý là sự xuất hiện của một số sản phẩm có giá cao nhưng thuế thấp và ngược lại. Biểu đồ cho thấy một hệ thống thuế phức tạp với nhiều mức thuế khác nhau áp dụng cho các sản phẩm, dịch vụ.

2.2.9 Biểu đồ tỷ lệ số lượng xe theo khoảng giá

```
import pandas as pd
import matplotlib.pyplot as plt
bins = [0, 2000, 4000, 6000, 8000, 10000, 12000, 14000, 16000,
float('inf')]
labels = ['<= 2000', '2001-4000', '4001-6000', '6001-8000', '
8001-10000', '10001-12000', '12001-14000', '14001-16000', '>
16000']
df['price_bin'] = pd.cut(df['price'], bins=bins, labels=labels
, right=False)
price_counts = df['price_bin'].value_counts()
colors = ['#A78FE7', '#7A28CC', '#62ABF5', '#FF9999', '#B565A2
', '#5CAB89', '#C90066', '#FF6666', '#CC0000']
plt.figure(figsize=(10, 8))
plt.pie(price_counts, labels=price_counts.index, autopct='%1.1
f%%', startangle=140, colors=colors)
plt.title('Bieu do ty le so luong xe theo khoang gia')
plt.axis('equal')
plt.show()
```

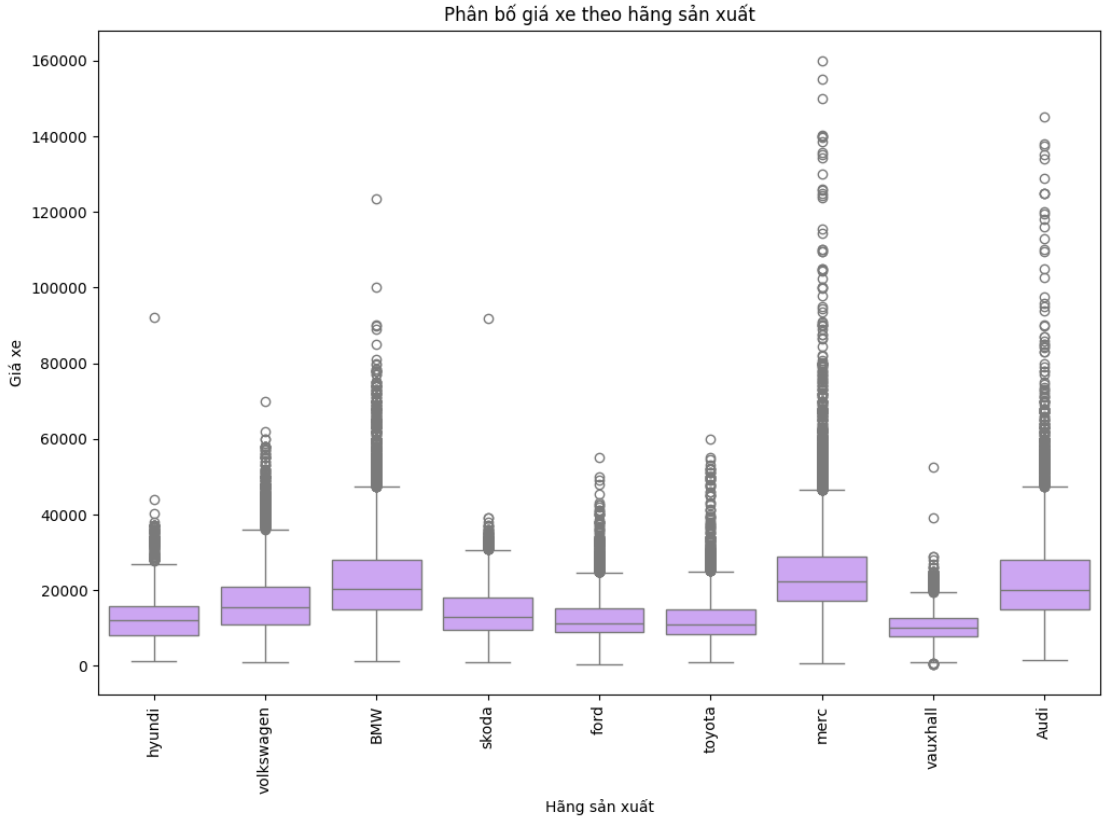


Hình 2.6. Biểu đồ tỷ lệ số lượng xe theo khoảng giá

Nhận xét: Nhìn chung phần lớn xe tập trung ở các khoảng giá trên 8000$, với đa số thuộc nhóm giá trên 16.000$. Điều này cho thấy thị trường xe hơi có xu hướng nghiêng về phân khúc các dòng xe trung và cao cấp.

2.2.10 Biểu đồ phân phối giá xe theo nhà sản xuất

```
    plt.figure(figsize=(12, 8))
    sns.boxplot(x='Manufacturer', y='price', data=df, color = '#
   CC99FF')
    plt.xticks(rotation=90)
    plt.title('Bieu do phan phoi gia xe theo nha san xuat')
    plt.xlabel('Nha san xuat')
    plt.ylabel('Gia xe')
    plt.show()
```

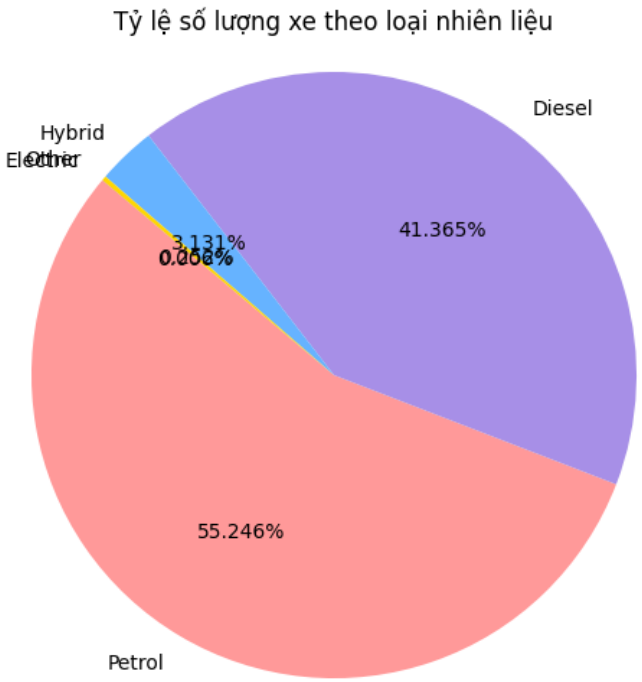


Hình 2.7. Biểu đồ phân phối giá xe theo nhà sản xuất

Nhận xét: Biểu đồ này thể hiện phân bố giá xe theo các hãng sản xuất khác nhau. BMW, Merc và Audi có phạm vi giá rộng nhất với nhiều xe cao cấp giá trên 100.000$. Vauxhall và Hyundai có phạm vi giá hẹp hơn, tập trung chủ yếu ở phân khúc giá thấp đến trung bình. Toyota và Ford có sự phân bố giá khá đồng đều còn Volkswagen và Skoda có phạm vi giá trung bình. Có một số điểm ngoại lệ ở hầu hết các hãng, thể hiện các mẫu xe đặc biệt hoặc cao cấp. Nhìn chung, biểu đồ cho thấy sự đa dạng về giá cả giữa các hãng xe, phản ánh chiến lược sản phẩm và phân khúc thị trường mục tiêu khác nhau của mỗi hãng.

2.2.11 Biểu đồ tỷ lệ số lượng xe theo loại nhiên liệu

```
import pandas as pd
import matplotlib.pyplot as plt
fuel_counts = df['fuelType'].value_counts()
colors = ['#FF9999','#A78FE7', '#66B3FF', '#FFD700', '#7A28CC'
]
plt.figure(figsize=(8, 6))
plt.pie(fuel_counts, labels=fuel_counts.index, autopct='%1.3f
%%', startangle=140, colors=colors)
plt.title('Bieu do ty le so luong xe theo loai nhien lieu')
plt.axis('equal')
plt.show()
```

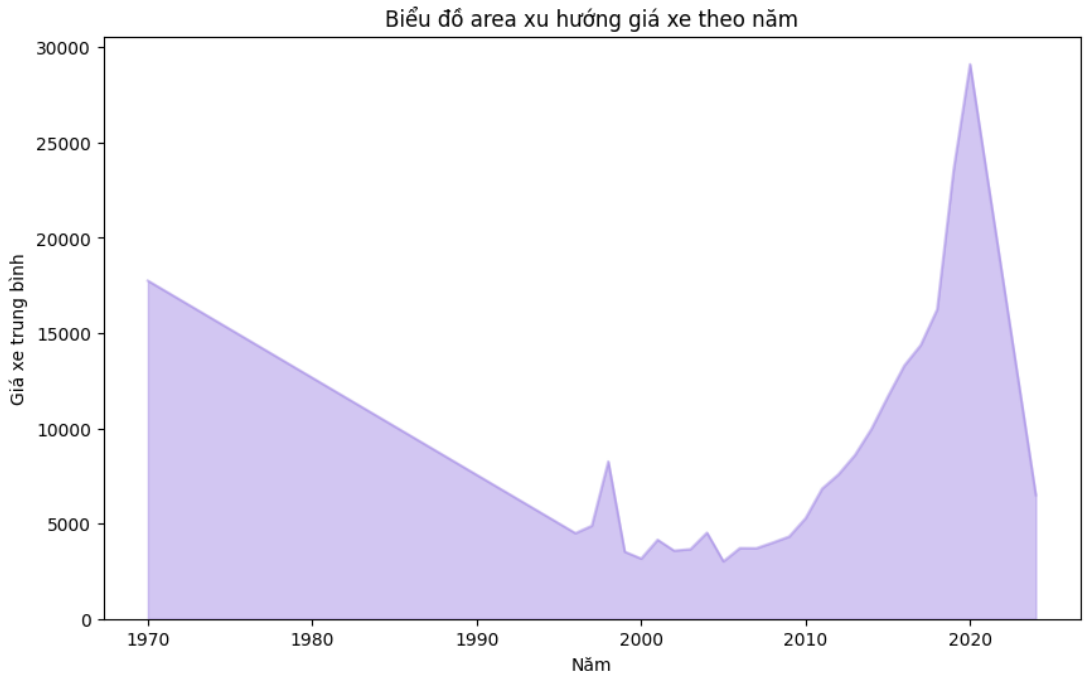


Hình 2.8. Biểu đồ tỷ lệ số lượng xe theo loại nhiên liệu

Nhận xét: Xe dùng nhiên liệu Petrol chiếm đa số với 55.246% tổng số xe. Xe dùng Diesel đứng thứ hai với 41.365%, theo sau là Hybrid với 3.131%. Các loại xe khác chiếm tỷ lệ không đáng kể. Điều này cho thấy xe dùng loại nhiên liệu Petrol và Diesel vẫn được dùng phổ biến nhất trên thị trường.

2.2.12 Biểu đồ Area xu hướng giá xe theo năm

```
import pandas as pd
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
df.groupby('year')['price'].mean().plot(kind='area', stacked=
False, color='#A78FE7')
plt.title('Bieu do Area xu huong gia xe theo nam')
plt.xlabel('Nam')
plt.ylabel('Gia xe trung binh')
plt.grid(False)
plt.show()
```



Hình 2.9. Biểu đồ Area xu hướng giá xe theo năm

Nhận xét: Biểu đồ thể hiện xu hướng giá xe theo thời gian từ 1970 đến 2020. Giá xe giảm dần từ 1970 đến khoảng năm 2000, sau đó có sự tăng nhẹ và dao động. Từ khoảng năm 2010, giá xe bắt đầu tăng mạnh và đạt đỉnh cao nhất vào năm 2020, cao hơn nhiều so với mức giá năm 1970. Xu hướng này cho thấy sự biến động lớn của thị trường xe trong 50 năm qua, với sự tăng giá đột biến trong thập kỷ gần đây.

2.3 Tiền xử lý dữ liệu

2.3.1 Loại bỏ khoảng trắng dư thừa trong cột Model

```
df['model'] = df['model'].str.strip()
```

2.3.2 Chuyển cột Price từ kiểu int sang kiểu float

```
df['price'] = df['price'].astype(float)
print(df.info())
```

Kết quả sau khi thực hiện:

```
Thông tin dữ liệu sau khi chuyển đổi cột 'price' sang float:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97712 entries, 0 to 97711
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   model         97712 non-null  object
 1   year          97712 non-null  int64
 2   price         97712 non-null  float64
 3   transmission  97712 non-null  object
 4   mileage       97712 non-null  int64
 5   fuelType      97712 non-null  object
 6   tax           97712 non-null  int64
 7   mpg           97712 non-null  float64
 8   engineSize    97712 non-null  float64
 9   Manufacturer  97712 non-null  object
dtypes: float64(3), int64(3), object(4)
memory usage: 7.5+ MB
None
```

Hình 2.10. Khám phá bộ dữ liệu

2.3.3 Chuyển đổi các dữ liệu định danh thành dạng số đối với các cột sau: Model, Transmission, FuelType và Manufacturer

Mã hóa số hóa là quá trình gán một số nguyên duy nhất cho mỗi giá trị duy nhất trong biến định danh. Các giá trị được gán các nhãn từ 0 đến N-1, trong đó N là số lượng giá trị duy nhất của biến.

```
    df3 = df.copy()
    categorical_features = ['model', 'transmission', 'fuelType', '
Manufacturer']

    label_encoders = {}
    for feature in categorical_features:
        label_encoders[feature] = LabelEncoder()
        df3[feature] = label_encoders[feature].fit_transform(df3[
feature])
```

2.3.4 Kiểm tra và xử lý dữ liệu ngoại lệ của các cột sau: Mileage, Mpg, EngineSize

Phương pháp Z-score là một kỹ thuật thống kê được sử dụng để đo độ lệch của một điểm dữ liệu so với trung bình của dữ liệu và đánh giá xem điểm đó có nằm ngoài phạm vi dự kiến hay không. Phương pháp này thường được áp dụng để phát hiện và xử lý các giá trị ngoại lệ trong dữ liệu.

Cách tính Z-score:

Cho một biến ngẫu nhiên X có giá trị x, Z-score của x, ký hiệu là Z, được tính theo công thức:

$$Z = \frac{x - \mu}{\sigma}$$

Trong đó:

- x là giá trị của biến cần tính Z-score.
- μ là kỳ vọng của biến X.
- σ là độ lệch chuẩn của biến X.

Ý nghĩa của Z-score:

Z-score biểu thị mức độ mà một giá trị cụ thể (hoặc điểm dữ liệu) khác biệt so với trung bình của dữ liệu, tính theo đơn vị độ lệch chuẩn. Giá trị Z-score dương cho thấy giá trị x lớn hơn trung bình μ, trong khi giá trị Z-score âm cho thấy x nhỏ hơn μ.

Phân phối Z-score:

Phân phối của Z-score theo giả thuyết là một phân phối chuẩn (normal distribution) với mean $\mu = 0$ và độ lệch chuẩn $\sigma = 1$. Do đó, giá trị Z-score càng xa khỏi 0 (ví dụ, vượt quá mức ± 3), càng cho thấy rằng giá trị x có thể được coi là ngoại lệ.

Ứng dụng của Z-score trong phát hiện ngoại lệ

Việc sử dụng Z-score để phát hiện ngoại lệ thường được thực hiện bằng cách:

1. Tính Z-score cho từng điểm dữ liệu trong tập dữ liệu.
2. Đặt một ngưỡng cho Z-score để xác định các điểm dữ liệu có Z-score vượt quá ngưỡng đó là ngoại lệ.
3. Loại bỏ hoặc xử lý các điểm dữ liệu được xác định là ngoại lệ.

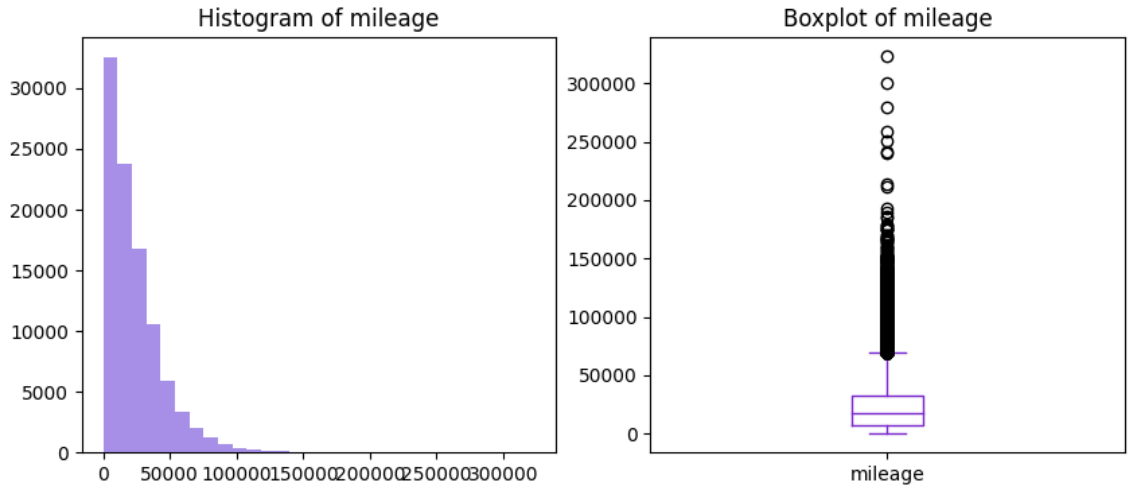
Mã nguồn cho thuật toán như sau:

```
df3 = df.copy()
z_scores = np.abs((df3 - df3.mean()) / df3.std())
outliers = (z_scores > 3).any(axis=1)
df3 = df3[~outliers]
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 92780 entries, 0 to 97711
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   model         92780 non-null  int64
 1   year          92780 non-null  int64
 2   price         92780 non-null  int64
 3   transmission  92780 non-null  int64
 4   mileage       92780 non-null  int64
 5   fuelType      92780 non-null  int64
 6   tax           92780 non-null  int64
 7   mpg           92780 non-null  float64
 8   engineSize    92780 non-null  float64
 9   Manufacturer  92780 non-null  int64
dtypes: float64(2), int64(8)
memory usage: 7.8 MB
```

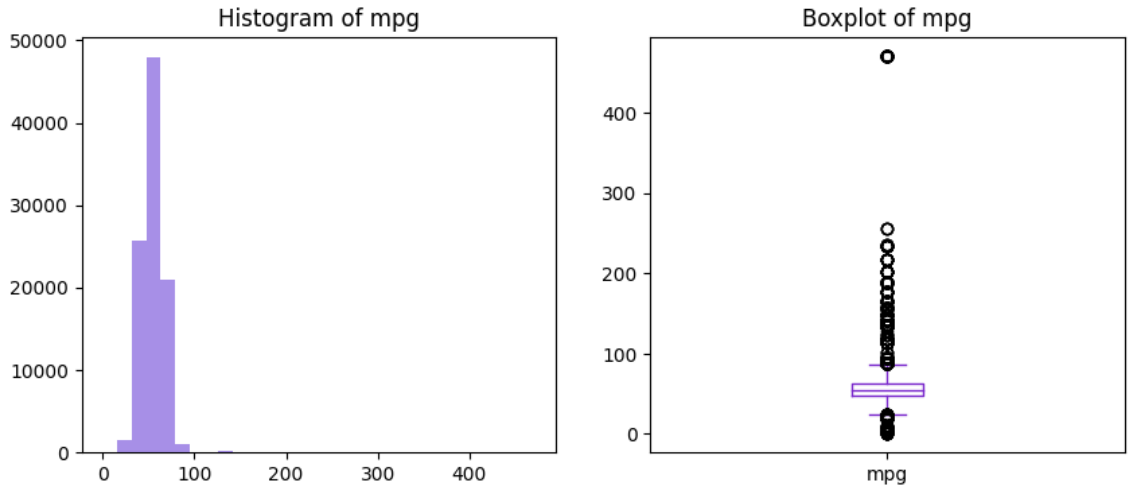
2.3.5 Biến đổi phân bố của một số cột

Phân bổ dữ liệu của "Mileage"của bộ dữ liệu ban đầu là:



Hình 2.11. Phân bổ dữ liệu của Mileage ban đầu

Phân bổ dữ liệu của "Mpg"của bộ dữ liệu ban đầu là:



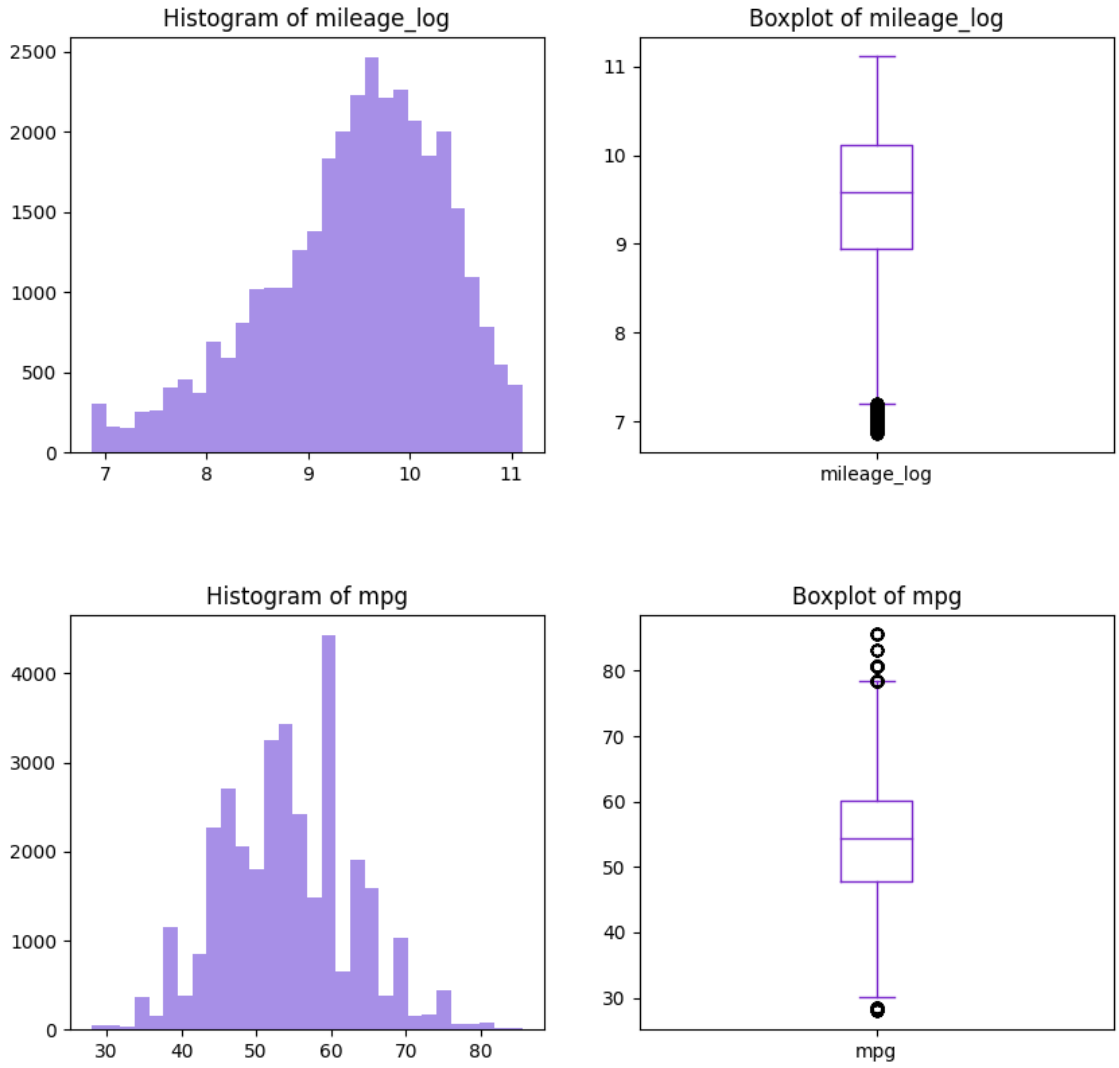
Hình 2.12. Phân bổ dữ liệu của MPG ban đầu

Nhận xét: Từ Histogram ta thấy dữ liệu bị lệch trái (có điểm ngoại lệ lệch nhiều về bên trái, hoặc "đuôi" của histogram nằm ở bên phải). Từ Boxplot ta thấy có khá nhiều điểm được coi là ngoại lệ. Các điểm ngoại lệ có thể được xử lý bằng cách clip về giá trị cực tiểu và cực đại của Box plot.

Clip dữ liệu theo Boxplot:

```
df3['mileage_log'] = np.log1p(df3['mileage'])
Q1 = df3.quantile(0.25)
Q3 = df3.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df2 = df3[(df3 >= lower_bound) & (df3 <= upper_bound)].dropna
()
```

Đoạn code sử dụng Biến đổi Logarithm tự nhiên cho dữ liệu thường được áp dụng để làm giảm độ lớn của dữ liệu và chuẩn hóa phân phối của nó. Sử dụng IQR để loại bỏ các điểm dữ liệu ngoại lai giúp làm sạch dữ liệu, làm giảm sự ảnh hưởng của các giá trị bất thường lên các phân tích và mô hình học máy sau này. Kết quả sau khi áp dụng:



Hình 2.13. Phân bổ dữ liệu Mileage và MPG sau khi biến đổi

3 Tạo, luyện và đánh giá mô hình

3.1 Tạo mô hình

3.1.1 Thuật toán kNN

a) Tổng quan về kNN

K-Nearest Neighbors (kNN) là một trong những thuật toán học có giám sát đơn giản nhất, thường được sử dụng trong khai phá dữ liệu và học máy. Thuật toán này không học bất kỳ điều gì từ tập dữ liệu huấn luyện, do đó kNN được xếp vào loại học lười biếng (lazy learning). Mọi tính toán chỉ được thực hiện khi cần dự đoán nhãn cho một dữ liệu mới.

Ý tưởng của kNN là dự đoán lớp (nhãn) của một đối tượng dữ liệu mới dựa trên các lớp (nhãn) của k hàng xóm gần nhất của nó.

K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. kNN còn được gọi là một thuật toán Instance-based hay Memory-based learning. Thuật toán kNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại gần nhau trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Việc tìm khoảng cách giữa 2 điểm củng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp.

b) Các bước thực hiện đối với kNN hồi quy

1. Giả sử D là tập các điểm dữ liệu đã được gắn nhãn. Mỗi điểm dữ liệu bao gồm các thuộc tính (đặc trưng) và một nhãn lớp tương ứng. A là dữ liệu mới chưa được phân loại mà bạn muốn dự đoán nhãn lớp cho nó.
2. Đo khoảng cách: Tính toán khoảng cách từ điểm dữ liệu mới A đến tất cả các điểm dữ liệu trong D. Các phương pháp đo khoảng cách phổ biến bao gồm:
   - Khoảng cách Euclidean: $d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
   - Khoảng cách Manhattan: $d(p,q) = \sum_{i=1}^{n}|p_i - q_i|$
   - Khoảng cách Minkowski: $d(p,q) = (\sum_{i=1}^{n}|p_i - q_i|^p)^{\frac{1}{p}}$
3. Xác định giá trị k, là số điểm dữ liệu lân cận bạn muốn xem xét. Lọc ra k điểm dữ liệu trong D có khoảng cách nhỏ nhất đến điểm dữ liệu mới A.

4. Tính toán giá trị dự đoán và gán giá trị dự đoán cho dữ liệu mới:
Sử dụng các giá trị liên tục của k điểm gần nhất để tính toán giá trị dự đoán cho điểm dữ liệu mới A. Có một số cách để tính toán:

- Trung bình cộng: Giá trị dự đoán là trung bình cộng của các giá trị liên tục của k điểm gần nhất.

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$$

- Trung bình có trọng số: Giá trị dự đoán là trung bình có trọng số của các giá trị liên tục của k điểm gần nhất, trong đó các trọng số thường là nghịch đảo của khoảng cách. (Trong đó y_i là giá trị liên tục của điểm dữ liệu thứ i và d_i là khoảng cách đến điểm dữ liệu thứ i).

$$\hat{y} = \frac{\sum_{i=1}^{k} \frac{y_i}{d_i}}{\sum_{i=1}^{k} \frac{1}{d_i}}$$

3.1.2 Khởi tạo mô hình

```
from sklearn.neighbors import KNeighborsRegressor
knn= KNeighborsRegressor()
```

3.2 Mô tả quá trình luyện mô hình hay chạy giải thuật

1. Chọn các thuộc tính còn lại để huấn luyện mô hình

```
features = ['model', 'year', 'transmission', 'mileage', 'mpg',
    'engineSize', 'Manufacturer', 'tax', 'fuelType']
X = df2[features]
y = df2['price']
```

2. Phân chia dữ liệu:
Tiếp theo tiến hành phân chia tập train và tập test theo tỷ lệ 80/20 nghĩa là tập train chiếm 80% trên tổng bộ dữ liệu và tập test chiếm 20% còn lại, sau đó thực hiện chuẩn hoá dữ liệu trên hai tập mới này. Việc chia dữ liệu như vậy nhằm đánh giá chính xác hiệu suất mô hình trên dữ liệu mới, ngăn ngừa hiện tượng overfitting và đảm bảo tính khách quan trong quá trình đánh giá mô hình. Điều này giúp mô hình học máy tổng quát hóa tốt hơn trên dữ liệu chưa từng thấy.

```
# Phan chia du lieu
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2)
# Chuan hoa du lieu
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3. Thực hiện khởi tạo và huấn luyện mô hình bằng thuật toán kNN

```
# Khoi tao va huan luyen mo hinh kNN
knn = KNeighborsRegressor(n_neighbors=7)
knn.fit(X_train_scaled, y_train)
```

3.3 Mô tả điều kiện dừng

Trong phương pháp kNN (k-Nearest Neighbors) dùng để dự đoán giá xe, điều kiện dừng không giống như trong các thuật toán học máy khác như cây quyết định hay mạng nơ-ron. KNN là một thuật toán dựa trên khoảng cách và không có quá trình huấn luyện thực sự, mà nó chỉ dựa vào việc tính toán khoảng cách đến các điểm dữ liệu đã có để dự đoán.

Tóm lại, không có "điều kiện dừng" cụ thể cho KNN vì nó là một thuật toán dựa trên tìm kiếm khoảng cách trong không gian đặc trưng. Thay vào đó, cần tập trung vào việc chọn k phù hợp, chuẩn hóa dữ liệu và tối ưu hóa các tham số để đạt được độ chính xác tốt nhất trong dự đoán giá xe.

3.4 Đánh giá quá trình luyện hay chạy giải thuật, hiệu chỉnh tham số

3.4.1 Kết quả chạy mô hình với dữ liệu test

```
    # Danh gia mo hinh tren tap kiem tra
    y_pred = knn.predict(X_test_scaled)
    results = pd.DataFrame({'Gia that': y_test, 'Gia du doan':
   y_pred})
    print(f'Gia du doan cho tap test:\n{results.head(30)}')
```

```
Giá dự đoán cho tập test:
       Giá thật   Giá dự đoán
29488   25985.0  24545.285714
54876   13499.0  13268.714286
59553   10242.0  10909.857143
22103    8395.0   9665.000000
63931   15298.0  17676.285714
55467    6988.0   7442.571429
60037   20000.0  19728.000000
63148   12750.0  13338.285714
14466    8985.0  10176.000000
4111    11850.0  12930.285714
52828   21999.0  24324.285714
14223   12495.0  11606.000000
61036    7999.0   8233.714286
11283    8500.0   9097.428571
2779    10998.0  11052.142857
63342   10498.0  10444.571429
91364   19750.0  20116.571429
14436   11990.0  12022.428571
92488   20690.0  22546.714286
55161   12699.0  12523.142857
58448   17990.0  21260.714286
2977     7495.0   7848.142857
23546   15446.0  14726.285714
31313   16695.0  17216.714286
23254   17495.0  20053.000000
70164   10179.0  11128.571429
34737   17298.0  17407.285714
80912   14897.0  15079.428571
8687     7991.0   8374.857143
96453   19194.0  15151.285714
```

3.4.2 Hiệu chỉnh tham số k

Thuật toán kNN có thể hiệu chỉnh k để đạt được kết quả tốt hơn. Cụ thể, sau đây là quá trình hiệu chỉnh k, có thể tìm k tốt nhất bằng thuật toán như sau:

```
#Tim kiem k tot nhat voi GridSearchCV
param_grid = {'n_neighbors': np.arange(1, 31)}
grid_search = GridSearchCV(KNeighborsRegressor(), param_grid,
cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train_scaled, y_train)
print(f"Gia tri k tot nhat: {grid_search.best_params_['
n_neighbors']}")
# Su dung k tot nhat de huan luyen va khoi tao mo hinh
best_k = grid_search.best_params_['n_neighbors']
knn = KNeighborsRegressor(n_neighbors=best_k)
knn.fit(X_train_scaled, y_train)
```

```
Giá trị k tốt nhất: 5
Giá dự đoán cho tập test:
       Giá thật  Giá dự đoán
29488   25985.0      24360.6
54876   13499.0      13476.4
59553   10242.0      11097.0
22103    8395.0       9538.0
63931   15298.0      18548.8
55467    6988.0       7539.6
60037   20000.0      18720.2
63148   12750.0      12575.6
14466    8985.0      10687.6
4111    11850.0      12693.4
52828   21999.0      24257.0
14223   12495.0      11752.2
61036    7999.0       8177.4
11283    8500.0       9139.2
2779    10998.0      11258.0
63342   10498.0      10583.0
91364   19750.0      19865.2
14436   11990.0      12237.4
92488   20690.0      22485.8
55161   12699.0      13035.4
58448   17990.0      22372.8
2977     7495.0       8066.6
23546   15446.0      14949.2
31313   16695.0      17113.6
23254   17495.0      20396.2
70164   10179.0      11080.0
34737   17298.0      17512.6
80912   14897.0      15732.2
8687     7991.0       8184.8
96453   19194.0      14712.2
```

3.5 Lựa chọn các số đo đánh giá mô hình

3.5.1 Mean Squared Error (MSE - Sai số bình phương trung bình)

MSE tính toán trung bình của bình phương của sai số giữa giá trị dự đoán và giá trị thực tế. MSE càng thấp thì mô hình càng tốt.

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(y_i - \bar{y})^2$$

```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error tren tap kiem tra: {mse}')
```

3.5.2 R^2 Score (Coefficient of Determination)

R^2 Score là một độ đo khác để đánh giá mức độ phù hợp của mô hình hồi quy. Giá trị R^2 càng gần 1 thì mô hình càng phù hợp với dữ liệu và giá trị càng gần 0 thì mô hình càng kém phù hợp.

$$R^2 = 1 - \frac{SS_E}{SS_T} = 1 - \frac{\sum_{i=1}^{k}(y_i - \hat{y})^2}{\sum_{i=1}^{k}(y_i - \bar{y})^2}$$

```
r2 = r2_score(y_test, y_pred)
print(f'R-squared tren tap kiem tra: {r2}')
```

3.6 Đánh giá mô hình với dữ liệu test hoặc với các kỹ thuật khác

3.6.1 Đánh giá mô hình với dữ liệu test

Để đánh giá quá trình luyện tay hay chạy giải thuật, ta dùng các chỉ số đánh giá MSE và R^2:

```
r2 = r2_score(y_test, y_pred)
print(f'R-squared tren tap kiem tra: {r2}')
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error tren tap kiem tra: {mse}')
```

Sau khi chạy code, ta thu được kết quả như sau:

```
R-squared trên tập kiểm tra: 0.9113959464793764
Mean Squared Error trên tập kiểm tra: 1945113.775074041
```

Nhận xét:

- R-squared (R^2): Giá trị R-squared là 0.9113, tức là mô hình giải thích được khoảng 91.13% sự biến thiên của giá trị thực tế của xe trên tập kiểm tra. Đây là một kết quả rất tích cực, cho thấy mô hình có khả năng giải thích tốt một phần lớn sự biến động của dữ liệu. Mô hình có xu hướng phù hợp tốt với dữ liệu mới, tức là nó không chỉ học thuật toán mà còn có khả năng áp dụng vào dữ liệu ngoài tập huấn luyện.
- Khả năng dự đoán chính xác (Prediction Accuracy): Mặc dù MSE là 1945113.775, điều này chỉ ra rằng sai số dự đoán trung bình bình phương vẫn cao. Điều này có thể cho thấy mô hình vẫn còn một số lượng lớn các dự đoán sai lệch với giá trị thực tế, có thể do sự phân phối không đồng đều của các điểm dữ liệu hoặc do mô hình chưa tối ưu hoặc không phù hợp hoàn toàn với dữ liệu. Tuy nhiên, R^2 cao cho thấy

mô hình có thể dự đoán chính xác hơn 91% sự biến thiên của giá trị thực tế trên tập kiểm tra.

- Tiềm năng cải thiện: Mặc dù đã có kết quả tích cực, việc tiếp tục tinh chỉnh và cải thiện mô hình là cần thiết để giảm MSE và tăng độ chính xác của dự đoán. Các phương pháp như tinh chỉnh siêu tham số, sử dụng các biến đổi dữ liệu (preprocessing), hoặc sử dụng các mô hình hồi quy khác có thể được xem xét để cải thiện hiệu suất.

- Tóm lại, mô hình hiện tại đã cho thấy khả năng dự đoán tốt và có sự phù hợp tổng quát đối với dữ liệu mới, nhưng vẫn còn tiềm năng để cải thiện và tối ưu hóa để đạt được các dự đoán chính xác hơn nữa.

3.6.2 Decision Tree Regressor (Hồi quy cây quyết định)

a) Thuật toán Decision Tree Regressor

Cây quyết định (Decision Tree)

Cây quyết định là một cấu trúc cây có thể được sử dụng để ra quyết định dựa trên các đặc trưng của dữ liệu. Mỗi nút trên cây đại diện cho một đặc trưng, mỗi nhánh từ nút này đại diện cho một giá trị của đặc trưng đó và mỗi lá của cây đại diện cho một dự đoán hoặc một nhóm.

Hồi quy cây quyết định (Decision Tree Regression)

Trong hồi quy cây quyết định, mục tiêu là dự đoán một giá trị liên tục (như giá xe) dựa trên các đặc trưng của các mẫu dữ liệu. Thuật toán hoạt động bằng cách phân chia không gian đặc trưng thành các vùng con (regions) sao cho mỗi vùng con có thể được ước tính một giá trị đầu ra.

Các bước chính trong thuật toán hồi quy cây quyết định:

1. Phân chia (Splitting): Thuật toán bắt đầu với một nút gốc (root node) chứa toàn bộ tập huấn luyện. Tại mỗi nút, thuật toán lựa chọn một đặc trưng và một ngưỡng để phân chia tập dữ liệu thành hai phần con. Mục tiêu là để tạo ra các phần con sao cho sai số (độ sai lệch giữa giá trị dự đoán và giá trị thực tế) giảm đi nhiều nhất có thể sau mỗi lần phân chia.

2. Xây dựng cây (Tree Building): Thuật toán tiếp tục phân chia tập dữ liệu theo các đặc trưng và ngưỡng cho đến khi đạt được điều kiện dừng, ví dụ như đạt đến một độ sâu cố định hoặc không thể giảm sai số một cách đáng kể thông qua phân chia tiếp theo.

3. Dừng (Stopping Criteria): Có nhiều tiêu chí dừng có thể được áp dụng, bao gồm độ

sâu của cây, số lượng mẫu tối thiểu trong mỗi lá, hoặc mức độ giảm sai số không đủ lớn sau mỗi phân chia.

4. Dự đoán (Prediction) Khi cây được xây dựng hoàn chỉnh, mô hình có thể dự đoán giá trị của một mẫu mới bằng cách đi từ nút gốc xuống các nút lá tương ứng với đặc trưng của mẫu đó.

b) Chạy mô hình với bộ dữ liệu train

Khởi tạo mô hình:

```
from sklearn.tree import DecisionTreeRegressor
tree_model = DecisionTreeRegressor()
```

Huấn luyện mô hình và dự đoán kết quả:

```
tree_model.fit(X_train_scaled, y_train)
y_pred = tree_model.predict(X_test_scaled)
results = pd.DataFrame({'Gia that': y_test, 'Gia du doan':
y_pred})
print(f'Gia xe du doan cho tap test':\n{results.head(30)}')
```

Giá dự đoán cho tập test:

	Giá thật	Giá dự đoán
29488	25985.0	22765.0
54876	13499.0	14691.0
59553	10242.0	10695.0
22103	8395.0	9970.0
63931	15298.0	19565.0
55467	6988.0	6750.0
60037	20000.0	18917.0
63148	12750.0	13995.0
14466	8985.0	10485.0
4111	11850.0	11999.0
52828	21999.0	23995.0
14223	12495.0	11990.0
61036	7999.0	8100.0
11283	8500.0	9499.0
2779	10998.0	10998.0
63342	10498.0	11990.0
91364	19750.0	17250.0
14436	11990.0	11995.0
92488	20690.0	23490.0
55161	12699.0	12600.0
58448	17990.0	19252.0
2977	7495.0	6940.0
23546	15446.0	15370.0
31313	16695.0	16775.0
23254	17495.0	18999.0
70164	10179.0	10491.0
34737	17298.0	17490.0
80912	14897.0	15995.0
8687	7991.0	7995.0
96453	19194.0	16991.0

Đánh giá mô hình:

```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error tren tap kiem tra: {mse}')
r2 = r2_score(y_test, y_pred)
print(f'R-squared tren tap kiem tra: {r2}')
```

```
Mean Squared Error trên tập kiểm tra: 2558434.928777564
R-squared trên tập kiểm tra: 0.8823307432225196
```

Nhận xét: Dựa trên cả hai chỉ số MSE và R^2, mô hình kNN thể hiện hiệu suất tốt hơn so với mô hình Decision Tree Regression trên tập kiểm tra:

- kNN có MSE thấp hơn, nghĩa là dự đoán của nó gần với giá trị thực tế hơn.
- kNN có R^2 cao hơn, nghĩa là nó giải thích được nhiều biến thiên trong dữ liệu hơn.

Do đó, mô hình kNN được đánh giá là tốt hơn so với mô hình Decision Tree Regression- trong ngữ cảnh của tập dữ liệu kiểm tra hiện tại.

4 Ứng dụng mô hình

4.1 Mô tả ứng dụng mô hình

Dự đoán giá xe với các xe có các yếu tố cụ thể như sau:

```
    new_data = pd.DataFrame({
        'model': [' Fiesta', ' Fiesta', ' 3 Series'],
        'year': [2019, 2017, 2017],
        'transmission': ['Manual', 'Automatic', 'Manual'],
        'mileage': [19910, 24468, 10586],
        'fuelType': [ 'Petrol', 'Petrol', 'Diesel'],
        'tax': [145, 150, 50],
        'mpg': [ 61.4, 54.3, 50.4],
        'engineSize': [1.0, 2.0, 1.2],
        'Manufacturer': ['ford', 'ford', 'BMW']
    })
    for feature in categorical_features:
        new_data[feature] = label_encoders[feature].transform(
   new_data[feature])

    new_data['mileage_log'] = np.log1p(new_data['mileage'])
    new_data_scaled = scaler.transform(new_data[features])
    # Du doan gia xe moi
    predicted_price = knn.predict(new_data_scaled)
    print(f'Gia du doan cho xe moi: {predicted_price}')
```

Kết quả dự đoán giá xe như sau:

```
Giá dự đoán cho xe mới: [16938. 14562. 11477.]
```

4.2 Diễn giải kết quả

1. Ford Fiesta 2019 (Manual, Petrol) có giá dự đoán: 16938

Với đặc trưng là xe năm 2019, hộp số tay (Manual), nhiên liệu xăng (Petrol) và các thông số kỹ thuật khác như mileage, tax, mpg và engineSize, mô hình đã dự đoán rằng giá của xe này vào khoảng 16938 đơn vị tiền tệ.

2. Ford Fiesta 2017 (Automatic, Petrol) có giá dự đoán: 14562
Đây là một phiên bản cũ hơn của Ford Fiesta so với chiếc đầu tiên. Với hộp số tự động (Automatic) và các đặc trưng khác, giá dự đoán là 14562 đơn vị tiền tệ. Giá thấp hơn so với chiếc Fiesta 2019 do tuổi đời và số kilomet đã đi nhiều hơn.

3. BMW 3 Series 2017 (Manual, Diesel) có giá dự đoán: 11477
Là một mẫu xe cao cấp hơn (BMW) nhưng có tuổi đời cũ hơn và chạy bằng diesel. Giá dự đoán là 11477 đơn vị tiền tệ. Mặc dù là BMW, nhưng tuổi đời và các đặc điểm kỹ thuật khác khiến giá của nó không cao bằng các mẫu Fiesta mới hơn.

5 Kết luận

5.1 Ưu nhược điểm của cách tiếp cận

Ưu điểm:

- KNN là một thuật toán rất dễ hiểu và triển khai. Đối với những người mới bắt đầu học máy, KNN là một công cụ tốt để làm quen với các khái niệm cơ bản.
- KNN hoạt động tốt với các bộ dữ liệu nhỏ và có số lượng đặc trưng ít, có thể cung cấp các dự đoán chính xác mà không cần phải điều chỉnh nhiều.
- Có thể được sử dụng cho cả bài toán phân loại (Classification) và bài toán hồi quy (Regression), làm cho nó trở nên đa năng trong nhiều ngữ cảnh khác nhau.
- Với các bộ dữ liệu đa chiều không quá lớn, KNN vẫn có thể hoạt động hiệu quả.
- Có thể làm việc với dữ liệu thuộc nhiều dạng khác nhau, từ số liên tục đến số rời rạc và thậm chí cả dữ liệu không theo cấu trúc, miễn là có thể xác định được khoảng cách giữa các điểm dữ liệu.
- Với sự phát triển của công nghệ và các kỹ thuật tối ưu hóa, KNN có thể được mở rộng để xử lý các bộ dữ liệu lớn hơn và tốc độ tính toán nhanh hơn.

Nhược điểm:

- Chi phí tính toán cao: Khi số lượng dữ liệu lớn, việc tính toán khoảng cách tới tất cả các điểm dữ liệu sẽ rất tốn kém về mặt thời gian và tài nguyên.
- Phụ thuộc vào lựa chọn k và khoảng cách: Hiệu suất của KNN phụ thuộc mạnh vào lựa chọn k và cách đo khoảng cách. Nếu không chọn đúng, kết quả có thể không chính xác.
- Khả năng bị nhiễu: KNN dễ bị ảnh hưởng bởi nhiễu (outliers) vì các điểm nhiễu có thể ảnh hưởng lớn tới kết quả dự đoán.
- HIệu suất không ổn định: Nếu dữ liệu có các đặc trưng với các thang đo khác nhau, các đặc trưng có giá trị lớn hơn có thể chi phối kết quả dự đoán. Do đó, dữ liệu cần được chuẩn hóa trước khi áp dụng KNN, điều này đòi hỏi thêm một bước xử lý dữ liệu.

5.2 Khả năng ứng dụng của kết quả nghiên cứu trong tương lai

- Phát triển hệ thống định giá xe tự động: Với các mô hình dự đoán giá xe, các công ty có thể phát triển hệ thống định giá tự động cho thị trường xe cũ. Điều này sẽ giúp khách hàng có cái nhìn rõ ràng và minh bạch về giá trị của xe.
- Ứng dụng trong các nền tảng thương mại điện tử: Các trang web bán xe trực tuyến có thể tích hợp các mô hình này để cung cấp cho người dùng các dự đoán về giá cả dựa trên các thông số kỹ thuật của xe.
- Phân tích thị trường và dự báo xu hướng: Kết quả nghiên cứu có thể được sử dụng để phân tích xu hướng giá xe theo thời gian, từ đó giúp các nhà sản xuất và đại lý xe đưa ra các chiến lược kinh doanh hiệu quả.
- Hỗ trợ quyết định mua bán: Các mô hình này có thể hỗ trợ người mua và người bán trong việc đưa ra quyết định dựa trên các dự đoán chính xác về giá trị của xe.
- Cải thiện dịch vụ bảo hiểm xe: Các công ty bảo hiểm có thể sử dụng các mô hình dự đoán giá để xác định giá trị bảo hiểm xe cũ, giúp việc đánh giá rủi ro trở nên chính xác hơn.

Tài liệu

[1] Car price, url: https://www.kaggle.com/datasets/meruvulikith/90000-cars-data-from-1970-to-2024/data, 2024.

[2] TS. Lê Hải Hà, *Bài giảng Hệ hỗ trợ quyết định*, Đại học Bách khoa Hà Nội, 2024.

[3] Machine Learning cho dữ liệu dạng bảng, url: https://machinelearningcoban.com/tabml_book/ch_embedding/embedding.html