

# Car Price Prediction

Car price prediction is based on finance and the marketing domain. It is a major research topic in machine learning because the price of a car depends on many factors. Some of the factors that contribute a lot to the price of a car are:

Brand Model Horsepower Mileage Safety Features GPS and many more.

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

data = pd.read_csv("CarPrice.csv")
data.head()
```

```
Out[2]:
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engine	location	whe
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd		front	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd		front	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd		front	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd		front	
4	5	2	audi 100ls	gas	std	four	sedan	4wd		front	

5 rows × 26 columns

```
In [3]: data.isnull().sum()
```

```
Out[3]: car_ID      0
        symboling  0
        CarName    0
        fueltype   0
        aspiration  0
        doornumber  0
        carbody     0
        drivewheel  0
        enginelocation 0
        wheelbase   0
        carlength   0
        carwidth    0
        carheight   0
        curbweight  0
        enginetype  0
        cylindernumber 0
        enginesize   0
        fuelsystem   0
        boreratio    0
        stroke       0
        compressionratio 0
        horsepower   0
        peakrpm      0
        citympg      0
        highwaympg   0
        price        0
        dtype: int64
```

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   car_ID                205 non-null   int64
 1   symboling              205 non-null   int64
 2   CarName                205 non-null   object
 3   fueltype               205 non-null   object
 4   aspiration              205 non-null   object
 5   doornumber             205 non-null   object
 6   carbody                205 non-null   object
 7   drivewheel             205 non-null   object
 8   enginelocation         205 non-null   object
 9   wheelbase              205 non-null   float64
10   carlength              205 non-null   float64
11   carwidth               205 non-null   float64
12   carheight              205 non-null   float64
13   curbweight             205 non-null   int64
14   enginetype             205 non-null   object
15   cylindernumber         205 non-null   object
16   enginesize             205 non-null   int64
17   fuelsystem             205 non-null   object
18   boreratio              205 non-null   float64
19   stroke                 205 non-null   float64
20   compressionratio       205 non-null   float64
21   horsepower             205 non-null   int64
22   peakrpm                205 non-null   int64
23   citympg                205 non-null   int64
24   highwaympg            205 non-null   int64
25   price                  205 non-null   float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
In [5]: print(data.describe())
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	\
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	

	curbweight	enginesize	boreratio	stroke	compressionratio	\
count	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	2555.565854	126.907317	3.329756	3.255415	10.142537	
std	520.680204	41.642693	0.270844	0.313597	3.972040	
min	1488.000000	61.000000	2.540000	2.070000	7.000000	
25%	2145.000000	97.000000	3.150000	3.110000	8.600000	
50%	2414.000000	120.000000	3.310000	3.290000	9.000000	
75%	2935.000000	141.000000	3.580000	3.410000	9.400000	
max	4066.000000	326.000000	3.940000	4.170000	23.000000	

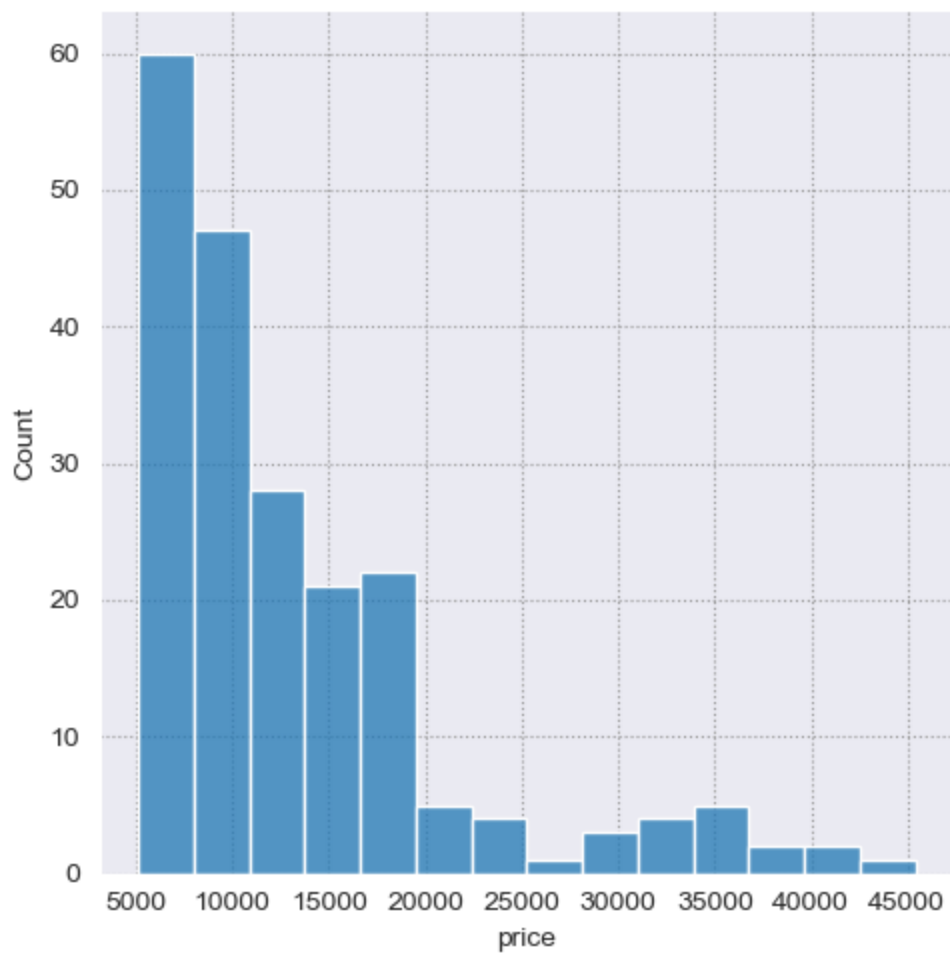
	horsepower	peakrpm	citympg	highwaympg	price
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	104.117073	5125.121951	25.219512	30.751220	13276.710571
std	39.544167	476.985643	6.542142	6.886443	7988.852332
min	48.000000	4150.000000	13.000000	16.000000	5118.000000
25%	70.000000	4800.000000	19.000000	25.000000	7788.000000
50%	95.000000	5200.000000	24.000000	30.000000	10295.000000
75%	116.000000	5500.000000	30.000000	34.000000	16503.000000
max	288.000000	6600.000000	49.000000	54.000000	45400.000000

```
In [6]: data.CarName.unique()
```

```
Out[6]: array(['alfa-romero giulia', 'alfa-romero stelvio',
'alfa-romero Quadrifoglio', 'audi100 ls', 'audi 100ls',
'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
'dodge rampage', 'dodge challenger se', 'dodge d200',
'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
'dodge coronet custom', 'dodge dart custom',
'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
'honda accord', 'honda civic 1300', 'honda prelude',
'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
'buick electra 225 custom', 'buick century luxus (sw)',
'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
'buick skylark', 'buick century special',
'buick regal sport coupe (turbo)', 'mercury cougar',
'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
'nissan latiao', 'nissan titan', 'nissan leaf', 'nissan juke',
'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
'peugeot 505s turbo diesel', 'plymouth fury iii',
'plymouth cricket', 'plymouth satellite custom (sw)',
'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
'porsche macan', 'porsche panamera', 'porsche cayenne',
'porsche boxer', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
'toyota corolla 1200', 'toyota corona hardtop',
'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
'toyota corolla', 'toyota corolla liftback',
'toyota celica gt liftback', 'toyota corolla tercel',
'toyota corona liftback', 'toyota starlet', 'toyota tercel',
'toyota cressida', 'toyota celica gt', 'toyota tercel',
'volkswagen rabbit', 'volkswagen 1131 deluxe sedan',
'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

```
In [10]: sns.set_style("darkgrid", {"grid.color": ".6", "grid.linestyle": ":"})
plt.figure(figsize=(15, 10))
sns.displot(data.price)
plt.show()
```

<Figure size 1500x1000 with 0 Axes>



```
In [14]: print(data.corr())
```

	car_ID	1.000000	-0.151621	0.129729	0.170636	0.052387	\
	symboling	-0.151621	1.000000	-0.531954	-0.357612	-0.232919	
	wheelbase	0.129729	-0.531954	1.000000	0.874587	0.795144	
	carlength	0.170636	-0.357612	0.874587	1.000000	0.841118	
	carwidth	0.052387	-0.232919	0.795144	0.841118	1.000000	
	carheight	0.255960	-0.541038	0.589435	0.491029	0.279210	
	curbweight	0.071962	-0.227691	0.776386	0.877728	0.867032	
	enginesize	-0.033930	-0.105790	0.569329	0.683360	0.735433	
	boreratio	0.260064	-0.130051	0.488750	0.606454	0.559150	
	stroke	-0.160824	-0.008735	0.160959	0.129533	0.182942	
	compressionratio	0.150276	-0.178515	0.249786	0.158414	0.181129	
	horsepower	-0.015006	0.070873	0.353294	0.552623	0.640732	
	peakrpm	-0.203789	0.273606	-0.360469	-0.287242	-0.220012	
	citympg	0.015940	-0.035823	-0.470414	-0.670909	-0.642704	
	highwaympg	0.011255	0.034606	-0.544082	-0.704662	-0.677218	
	price	-0.109093	-0.079978	0.577816	0.682920	0.759325	

		carheight	curbweight	enginesize	boreratio	stroke	\
	car_ID	0.255960	0.071962	-0.033930	0.260064	-0.160824	
	symboling	-0.541038	-0.227691	-0.105790	-0.130051	-0.008735	
	wheelbase	0.589435	0.776386	0.569329	0.488750	0.160959	
	carlength	0.491029	0.877728	0.683360	0.606454	0.129533	
	carwidth	0.279210	0.867032	0.735433	0.559150	0.182942	
	carheight	1.000000	0.295572	0.067149	0.171071	-0.055307	
	curbweight	0.295572	1.000000	0.850594	0.648480	0.168790	
	enginesize	0.067149	0.850594	1.000000	0.583774	0.203129	
	boreratio	0.171071	0.648480	0.583774	1.000000	-0.055909	
	stroke	-0.055307	0.168790	0.203129	-0.055909	1.000000	
	compressionratio	0.261214	0.151362	0.028971	0.005197	0.186110	
	horsepower	-0.108802	0.750739	0.809769	0.573677	0.080940	
	peakrpm	-0.320411	-0.266243	-0.244660	-0.254976	-0.067964	
	citympg	-0.048640	-0.757414	-0.653658	-0.584532	-0.042145	
	highwaympg	-0.107358	-0.797465	-0.677470	-0.587012	-0.043931	
	price	0.119336	0.835305	0.874145	0.553173	0.079443	

		compressionratio	horsepower	peakrpm	citympg	\
	car_ID	0.150276	-0.015006	-0.203789	0.015940	
	symboling	-0.178515	0.070873	0.273606	-0.035823	
	wheelbase	0.249786	0.353294	-0.360469	-0.470414	
	carlength	0.158414	0.552623	-0.287242	-0.670909	
	carwidth	0.181129	0.640732	-0.220012	-0.642704	
	carheight	0.261214	-0.108802	-0.320411	-0.048640	
	curbweight	0.151362	0.750739	-0.266243	-0.757414	
	enginesize	0.028971	0.809769	-0.244660	-0.653658	
	boreratio	0.005197	0.573677	-0.254976	-0.584532	
	stroke	0.186110	0.080940	-0.067964	-0.042145	
	compressionratio	1.000000	-0.204326	-0.435741	0.324701	
	horsepower	-0.204326	1.000000	0.131073	-0.801456	
	peakrpm	-0.435741	0.131073	1.000000	-0.113544	
	citympg	0.324701	-0.801456	-0.113544	1.000000	
	highwaympg	0.265201	-0.770544	-0.054275	0.971337	
	price	0.067984	0.808139	-0.085267	-0.685751	

		highwaympg	price
	car_ID	0.011255	-0.109093
	symboling	0.034606	-0.079978
	wheelbase	-0.544082	0.577816
	carlength	-0.704662	0.682920
	carwidth	-0.677218	0.759325
	carheight	-0.107358	0.119336
	curbweight	-0.797465	0.835305
	enginesize	-0.677470	0.874145
	boreratio	-0.587012	0.553173

```

stroke            -0.043931  0.079443
compressionratio  0.265201  0.067984
horsepower        -0.770544  0.808139
peakrpm           -0.054275 -0.085267
citympg           0.971337 -0.685751
highwaympg        1.000000 -0.697599
price             -0.697599  1.000000

```

C:\Users\User\AppData\Local\Temp\ipykernel\_1700\1904353132.py:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

```
print(data.corr())
```

```

In [16]: plt.figure(figsize=(20, 15))
correlations = data.corr()
sns.heatmap(correlations, cmap="icefire", annot=True)
plt.show()

```

C:\Users\User\AppData\Local\Temp\ipykernel\_1700\743282463.py:2: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

```
correlations = data.corr()
```



```

In [17]: predict = "price"
data = data[["symboling", "wheelbase", "carlength",
            "carwidth", "carheight", "curbweight",
            "enginesize", "boreratio", "stroke",

```

```

        "compressionratio", "horsepower", "peakrpm",
        "citympg", "highwaympg", "price"]])
x = np.array(data.drop([predict], 1))
y = np.array(data[predict])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)

from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
predictions = model.predict(xtest)

from sklearn.metrics import mean_absolute_error
model.score(xtest, predictions)

```

C:\Users\User\AppData\Local\Temp\ipykernel\_1700\3945656543.py:7: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.

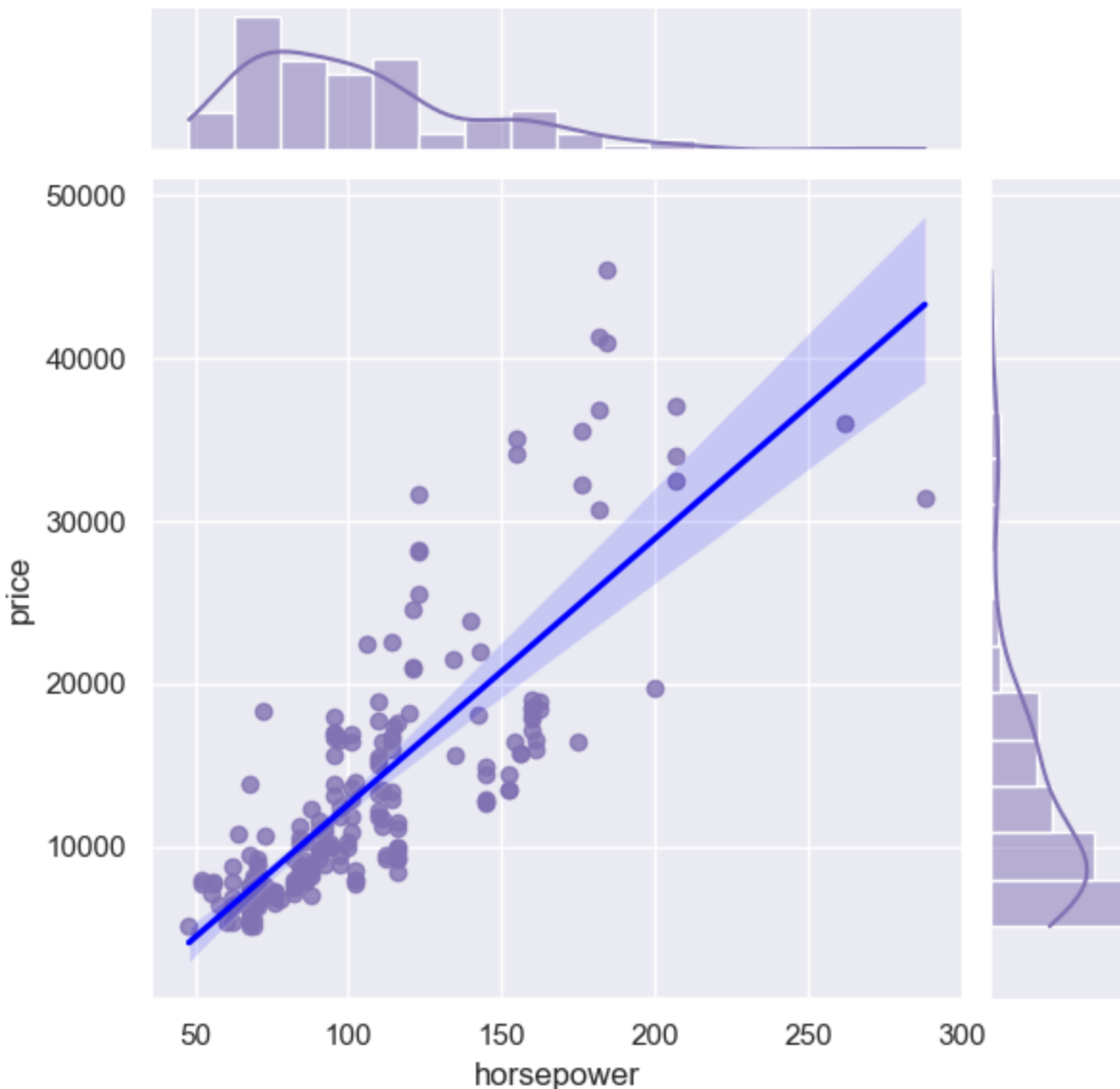
```
x = np.array(data.drop([predict], 1))
```

Out[17]: 1.0

```

In [20]: sns.set_theme(style="darkgrid")
sns.jointplot(x="horsepower", y="price", data=data,
              kind="reg", color="m", line_kws={'color': 'blue'})
plt.show()

```





In [ ]: