## IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

## IMPORTINNG DATASET

```
car_df = pd.read_csv('https://raw.githubusercontent.com/amankharwal/Website-data/master/CarPrice.csv')
car_df.head()
```

| | car_ID | symboling | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | enginelocation | wheelbase | ... | enginesize |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3 | alfa-romero giulia | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 |
| 1 | 2 | 3 | alfa-romero stelvio | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 |
| 2 | 3 | 1 | alfa-romero Quadrifoglio | gas | std | two | hatchback | rwd | front | 94.5 | ... | 152 |
| 3 | 4 | 2 | audi 100 ls | gas | std | four | sedan | fwd | front | 99.8 | ... | 109 |
| 4 | 5 | 2 | audi 100ls | gas | std | four | sedan | 4wd | front | 99.4 | ... | 136 |

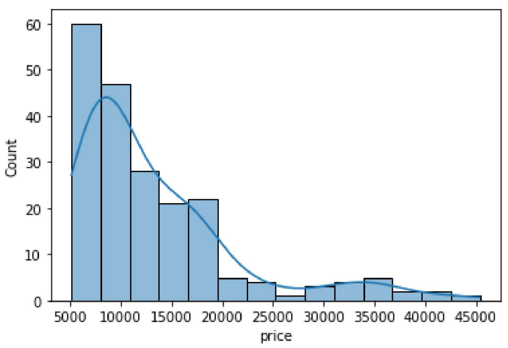
5 rows × 26 columns

## DATA ANALYSIS

```
car_df.info()
car_df.describe()
```

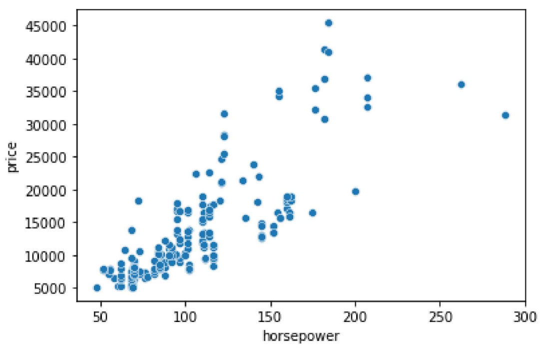
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
```

```
sns.histplot(car_df['price'], kde=True)
plt.show()
```

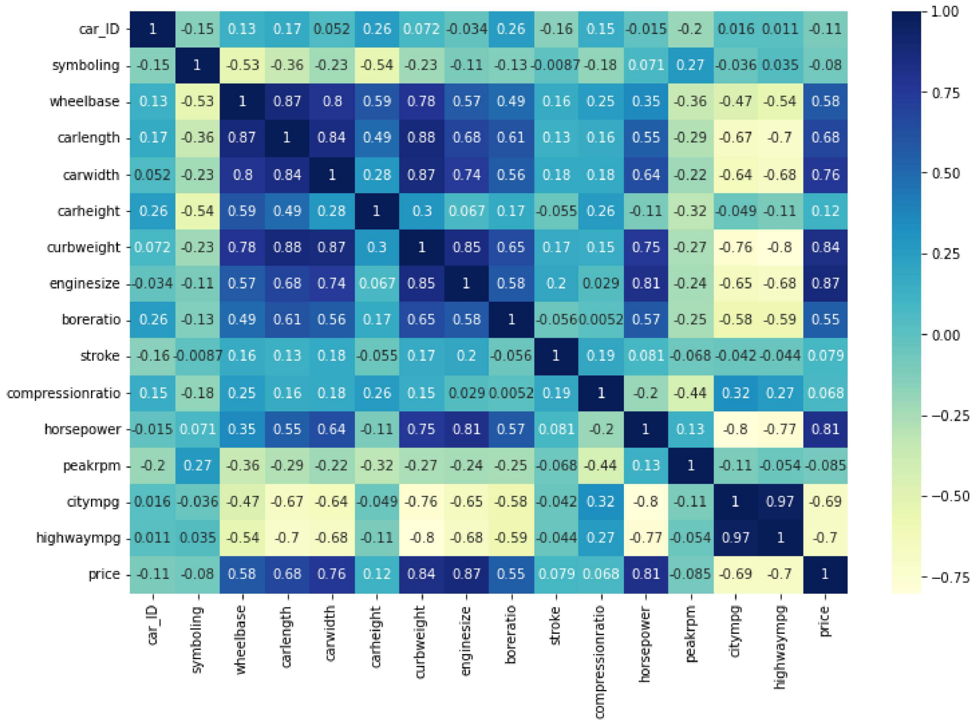



```
 24  highwaympg        205 non-null    int64
```

```
sns.scatterplot(x='horsepower', y='price', data=car_df)
plt.show()
```




```
plt.figure(figsize=(12, 8))
sns.heatmap(car_df.corr(), annot=True, cmap='YlGnBu')
plt.show()
```



## DATA PREPARATION

```
# removing irrelevant columns
car_df = car_df.drop(['CarName', 'car_ID'], axis=1)

# handling missing values
car_df.isnull().sum()

# handling categorical variables
car_df = pd.get_dummies(car_df, drop_first=True)
```

## CORRELATION TABLE

```
car_df.corr()
```

| | symboling | wheelbase | carlength | carwidth | carheight | curbweight | enginesize | bor |
|---|---|---|---|---|---|---|---|---|
| **symboling** | 1.000000 | -0.531954 | -0.357612 | -0.232919 | -0.541038 | -0.227691 | -0.105790 | -0 |
| **wheelbase** | -0.531954 | 1.000000 | 0.874587 | 0.795144 | 0.589435 | 0.776386 | 0.569329 | 0 |
| **carlength** | -0.357612 | 0.874587 | 1.000000 | 0.841118 | 0.491029 | 0.877728 | 0.683360 | 0 |
| **carwidth** | -0.232919 | 0.795144 | 0.841118 | 1.000000 | 0.279210 | 0.867032 | 0.735433 | 0 |
| **carheight** | -0.541038 | 0.589435 | 0.491029 | 0.279210 | 1.000000 | 0.295572 | 0.067149 | 0 |
| **curbweight** | -0.227691 | 0.776386 | 0.877728 | 0.867032 | 0.295572 | 1.000000 | 0.850594 | 0 |
| **enginesize** | -0.105790 | 0.569329 | 0.683360 | 0.735433 | 0.067149 | 0.850594 | 1.000000 | 0 |
| **boreratio** | -0.130051 | 0.488750 | 0.606454 | 0.559150 | 0.171071 | 0.648480 | 0.583774 | 1 |
| **stroke** | -0.008735 | 0.160959 | 0.129533 | 0.182942 | -0.055307 | 0.168790 | 0.203129 | -0 |
| **aspiration_turbo** | -0.059866 | 0.257611 | 0.234539 | 0.300567 | 0.087311 | 0.324902 | 0.108217 | 0 |

## SPLITTING DATA INTO TRAIN AND TEST DATASETS

```
from sklearn.model_selection import train_test_split

X = car_df.drop('price', axis=1)
y = car_df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

## LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
lr_model = LinearRegression()
```

## TRAINING AND EVALUATING THE MODEL

```
lr_model.fit(X_train, y_train)
y_pred = lr_model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 7052032.685191006
```

```
rmse = np.sqrt(mse)
print("\nRoot Mean Squared Error:", rmse)
```

```

Root Mean Squared Error: 2655.566358649508
```

```
r2 = r2_score(y_test, y_pred)
print("\nR-squared Score:", r2)
```

```

R-squared Score: 0.8982161227892841
```

```
print("\nACCURACY:",lr_model.score(X_test,y_test)*100)
```

```

ACCURACY: 89.82161227892841
```