car price prediction

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

In [2]:

```
df = pd.read_csv('CarPrice.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wh
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 [4]:

```
df.tail()
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	whe
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	

5 rows × 26 columns

In [5]:
```
df.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 [6]:
```
df.describe()
```

Out[6]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	bo
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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 [8]:
```
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
df.shape
```

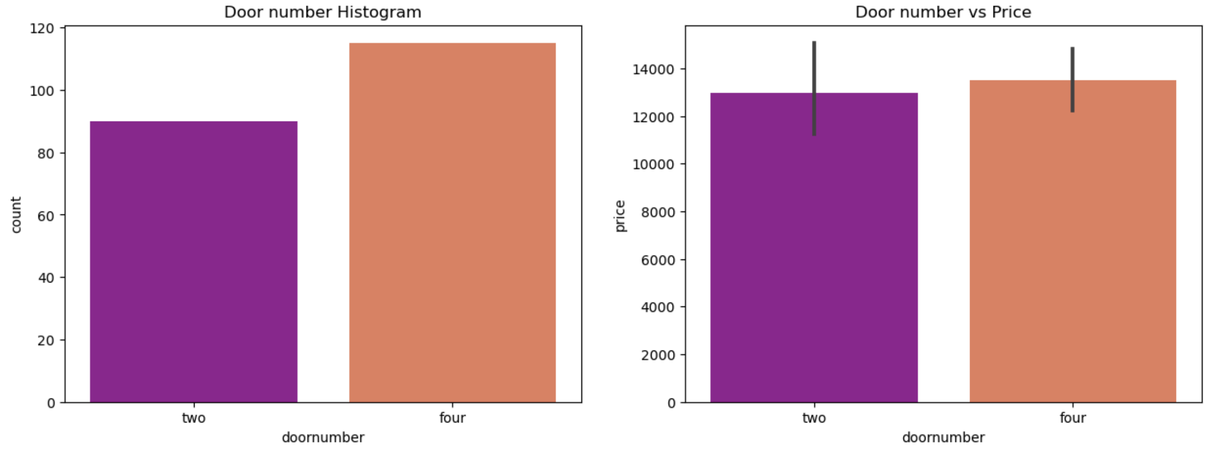
Out[9]:
```
(205, 26)
```

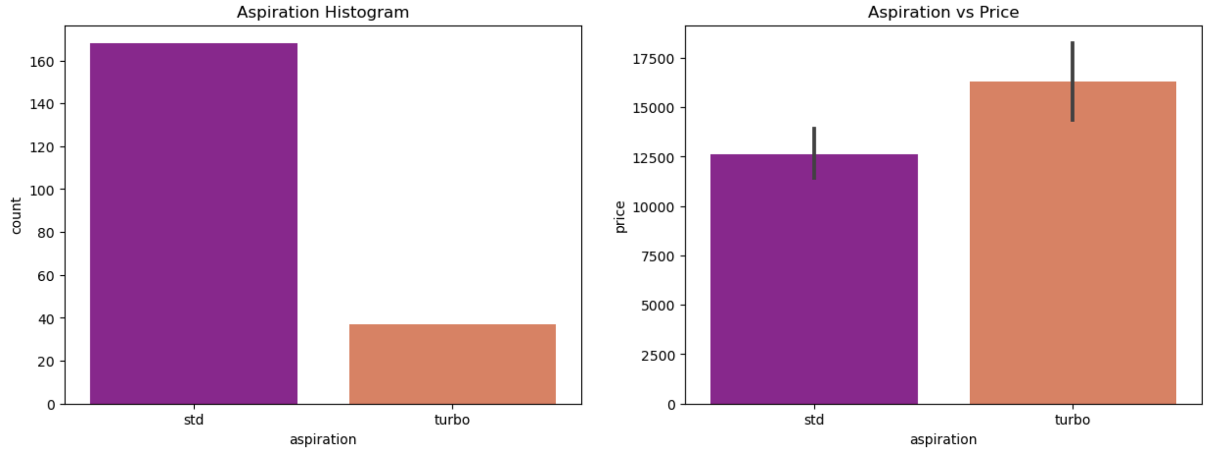
In [10]:
```
print(df.price.describe(percentiles=[0.225,0.50,0.75,0.85,0.98,1]))
```

```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
22.5%     7609.000000
50%      10295.000000
75%      16503.000000
85%      18500.000000
98%      36809.600000
100%     45400.000000
max      45400.000000
Name: price, dtype: float64
```

Data Visualzation

In [11]:
```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.title("Door number Histogram")
sns.countplot(data=df, x='doornumber', palette="plasma")
plt.subplot(1,2,2)
plt.title('Door number vs Price')
sns.barplot(data=df, x='doornumber', y='price', palette="plasma")
plt.show()

plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.title("Aspiration Histogram")
sns.countplot(data=df, x='aspiration', palette="plasma")
plt.subplot(1,2,2)
plt.title("Aspiration vs Price")
sns.barplot(data=df, x='aspiration', y='price', palette="plasma")
plt.show()
```



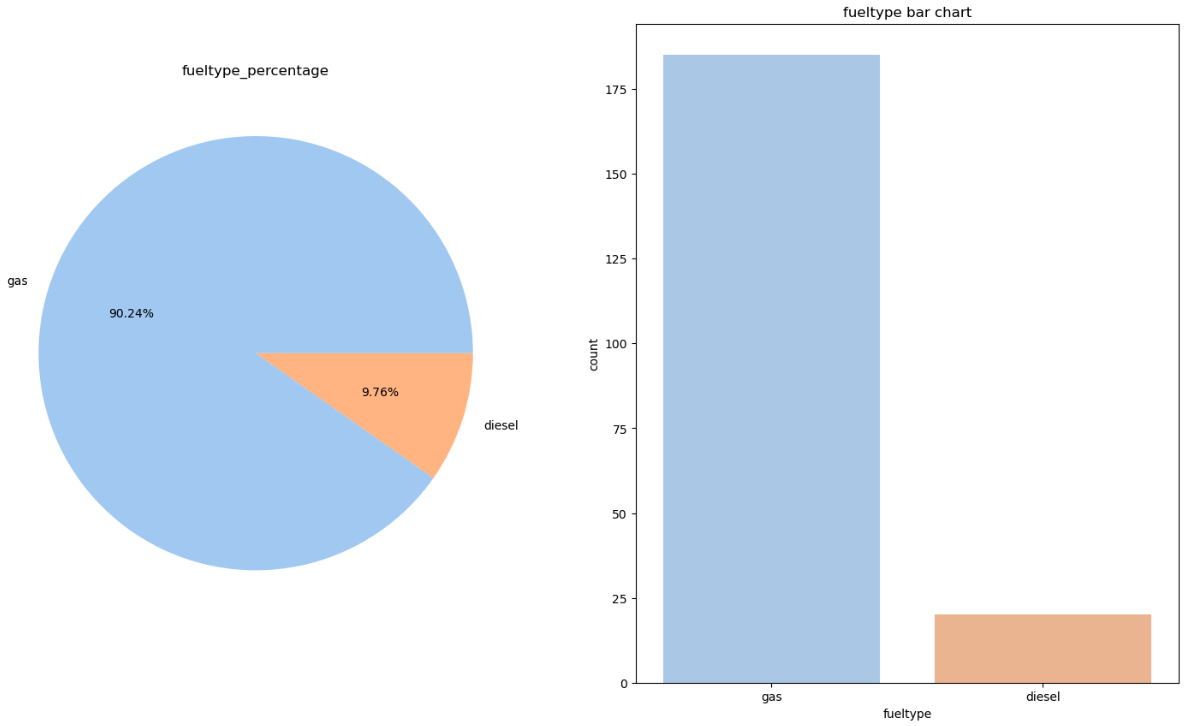


In [12]:

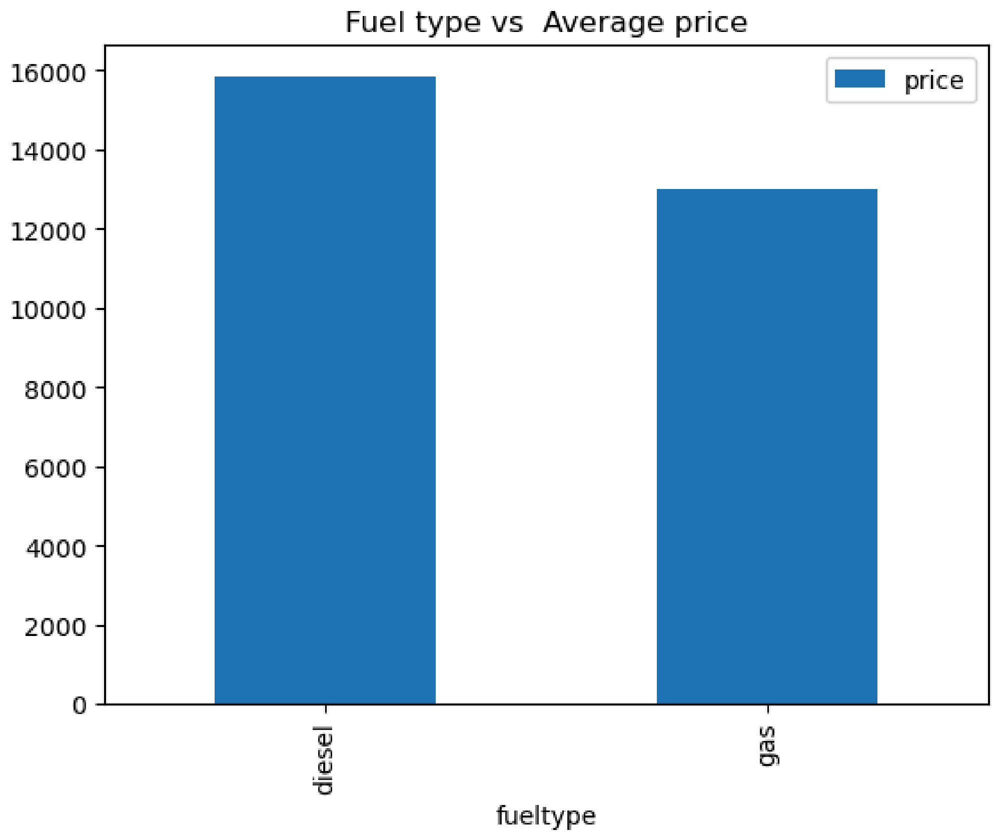
```
colors=sns.color_palette('pastel')
labels=df['fueltype'].dropna().unique()
plt.figure(figsize=(18,10))
plt.subplot(1,2,1)

plt.title('fueltype_percentage')
plt.pie(df['fueltype'].value_counts(),labels=labels,colors=colors,autopct='%.2f%%')
plt.subplot(1,2,2)
plt.title('fueltype bar chart')
sns.countplot(x='fueltype',data=df,palette=colors)
df.fueltype.value_counts(dropna=False)
```

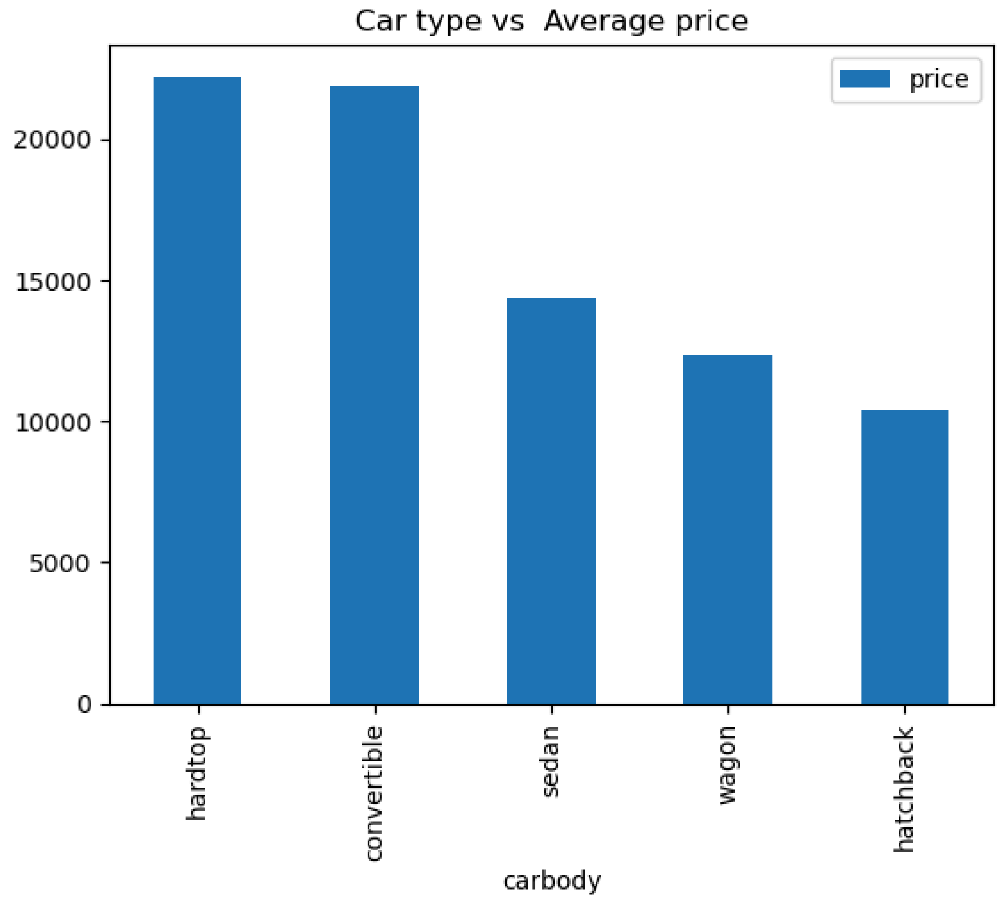
Out[12]:

```
fueltype
gas       185
diesel     20
Name: count, dtype: int64
```



In [13]:
```
dff=pd.DataFrame(df.groupby(['fueltype'])['price'].mean().sort_values(ascending=False))
dff.plot.bar()
plt.title("Fuel type vs  Average price")
plt.show()
dff=pd.DataFrame(df.groupby(['carbody'])['price'].mean().sort_values(ascending=False))
dff.plot.bar()
plt.title("Car type vs  Average price")
plt.show()
```





In [15]:

```
y=df['price']
x=df[['symboling','wheelbase','carwidth', 'carheight', 'curbweight', 'enginesize','borera
```

In [16]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100
```

Random Forest Model

In [17]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [18]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100
print('training data shape is:{}.'.format(x_train.shape))
print('training label shape is:{}.'.format(y_train.shape))
print('testing data shape is:{}.'.format(x_test.shape))
print('testing data shape is:{}.'.format(y_test.shape))
```

```
training data shape is:(164, 13).
training label shape is:(164,).
testing data shape is:(41, 13).
testing data shape is:(41,).
```

In [19]:

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
```

In [20]:
```
regressor.fit(x,y)
```

Out[20]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [21]:
```
regressor.score(x_train,y_train)
```

Out[21]:
```
0.9883097125128573
```

In [22]:
```
regressor.score(x_test,y_test)
```

Out[22]:
```
0.9867846662115134
```

In [23]:
```
from sklearn.metrics import accuracy_score
predictions=regressor.predict(x_test)
```

In [24]:
```
percentage=regressor.score(x_test,y_test)
percentage
```

Out[24]:
```
0.9867846662115134
```

In [25]:
```
print(regressor.score(x_train,y_train))
print(f"test set:{len(x_test)}")
print(f"Accuracy={percentage*100}%")
```

```
0.9883097125128573
test set:41
Accuracy=98.67846662115134%
```

Thank you

In [ ]:
```
```