

In [1]:

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 from sklearn import preprocessing,svm
6 from sklearn.model_selection import train_test_split
7 from sklearn.linear_model import LinearRegression
```

In [2]:

```
1 df=pd.read_csv(r"C:\Users\HP\Downloads\used_cars_data.csv")
2 df
```

Out[2]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	
...
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	

7253 rows × 14 columns



In [3]:

```
1 df.head()
```

Out[3]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	Fi
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	Fi
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	Fi
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	Fi
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Seco

In [4]:

```
1 df.describe()
```

Out[4]:

	S.No.	Year	Kilometers_Driven	Seats	Price
count	7253.000000	7253.000000	7.253000e+03	7200.000000	6019.000000
mean	3626.000000	2013.365366	5.869906e+04	5.279722	9.479468
std	2093.905084	3.254421	8.442772e+04	0.811660	11.187917
min	0.000000	1996.000000	1.710000e+02	0.000000	0.440000
25%	1813.000000	2011.000000	3.400000e+04	5.000000	3.500000
50%	3626.000000	2014.000000	5.341600e+04	5.000000	5.640000
75%	5439.000000	2016.000000	7.300000e+04	5.000000	9.950000
max	7252.000000	2019.000000	6.500000e+06	10.000000	160.000000

In [5]:

```
1 df.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   S.No.                 7253 non-null   int64  
 1   Name                  7253 non-null   object  
 2   Location              7253 non-null   object  
 3   Year                  7253 non-null   int64  
 4   Kilometers_Driven     7253 non-null   int64  
 5   Fuel_Type             7253 non-null   object  
 6   Transmission          7253 non-null   object  
 7   Owner_Type            7253 non-null   object  
 8   Mileage               7251 non-null   object  
 9   Engine                7207 non-null   object  
10   Power                 7207 non-null   object  
11   Seats                 7200 non-null   float64 
12   New_Price             1006 non-null   object  
13   Price                 6019 non-null   float64 
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

In [6]:

```
1 df.shape
```

Out[6]:

```
(7253, 14)
```

In [7]:

```
1 df.isna().any()
```

Out[7]:

```
S.No.          False
Name           False
Location       False
Year           False
Kilometers_Driven False
Fuel_Type      False
Transmission   False
Owner_Type     False
Mileage        True
Engine         True
Power          True
Seats          True
New_Price      True
Price          True
dtype: bool
```

In [8]:

```
1 df.isnull().sum()
```

Out[8]:

```
S.No.          0
Name           0
Location       0
Year           0
Kilometers_Driven  0
Fuel_Type      0
Transmission   0
Owner_Type     0
Mileage        2
Engine         46
Power          46
Seats          53
New_Price     6247
Price         1234
dtype: int64
```

In [9]:

```
1 df.fillna(value=0,inplace=True)
```

In [10]:

```
1 df.isnull().sum()
```

Out[10]:

```
S.No.          0
Name           0
Location       0
Year           0
Kilometers_Driven  0
Fuel_Type      0
Transmission   0
Owner_Type     0
Mileage        0
Engine         0
Power          0
Seats          0
New_Price     0
Price         0
dtype: int64
```

In [11]:

```
1 df=df[['Year','Price']]
2 df.columns=['yr','pr']
```

In [12]:

```
1 df.head(10)
```

Out[12]:

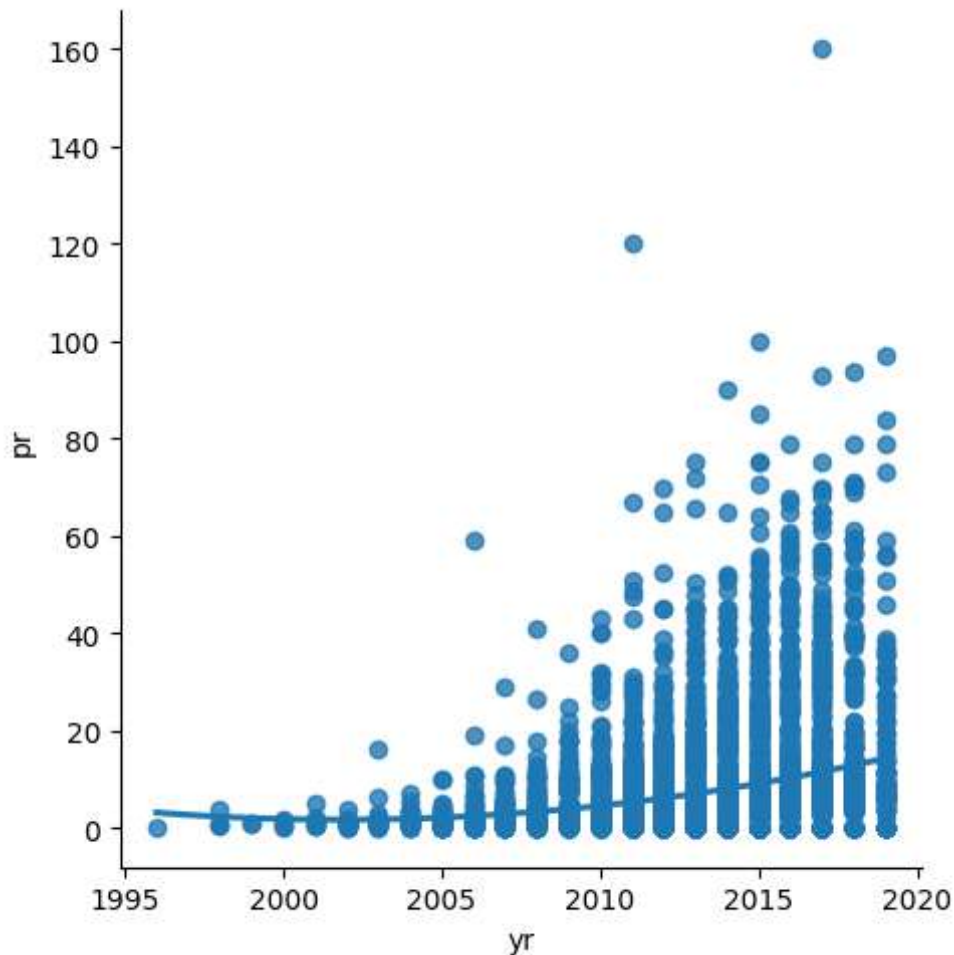
	yr	pr
0	2010	1.75
1	2015	12.50
2	2011	4.50
3	2012	6.00
4	2013	17.74
5	2012	2.35
6	2013	3.50
7	2016	17.50
8	2013	5.20
9	2012	1.95

In [13]:

```
1 sns.lmplot(x="yr",y="pr",data=df,order=2,ci=None)
```

Out[13]:

<seaborn.axisgrid.FacetGrid at 0x20fe80ae090>



In [14]:

```
1 df.describe()
```

Out[14]:

	yr	pr
count	7253.000000	7253.000000
mean	2013.365366	7.866665
std	3.254421	10.796286
min	1996.000000	0.000000
25%	2011.000000	2.290000
50%	2014.000000	4.650000
75%	2016.000000	8.400000
max	2019.000000	160.000000

In [15]:

```
1 df.fillna(method='ffill',inplace=True)
```

C:\Users\HP\AppData\Local\Temp\ipykernel_4624\4116506308.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df.fillna(method='ffill',inplace=True)
```

In [16]:

```
1 x=np.array(df['yr']).reshape(-1,1)
2 y=np.array(df['pr']).reshape(-1,1)
```

In [17]:

```
1 df.dropna(inplace=True)
```

C:\Users\HP\AppData\Local\Temp\ipykernel_4624\1379821321.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df.dropna(inplace=True)
```

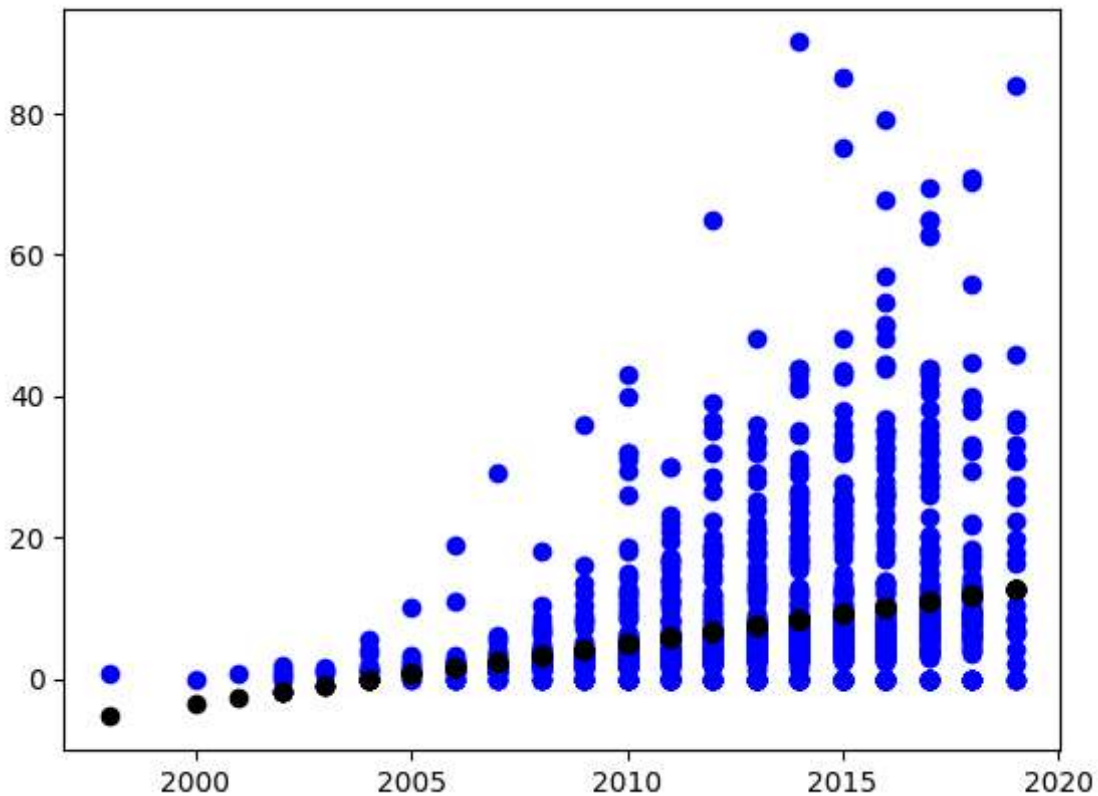
In [18]:

```
1 X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
2 regr=LinearRegression()
3 regr.fit(X_train,y_train)
4 print(regr.score(X_test,y_test))
```

0.07388164196776581

In [19]:

```
1 y_pred=regr.predict(X_test)
2 plt.scatter(X_test,y_test,color='b')
3 plt.scatter(X_test,y_pred,color='k')
4 plt.show()
```

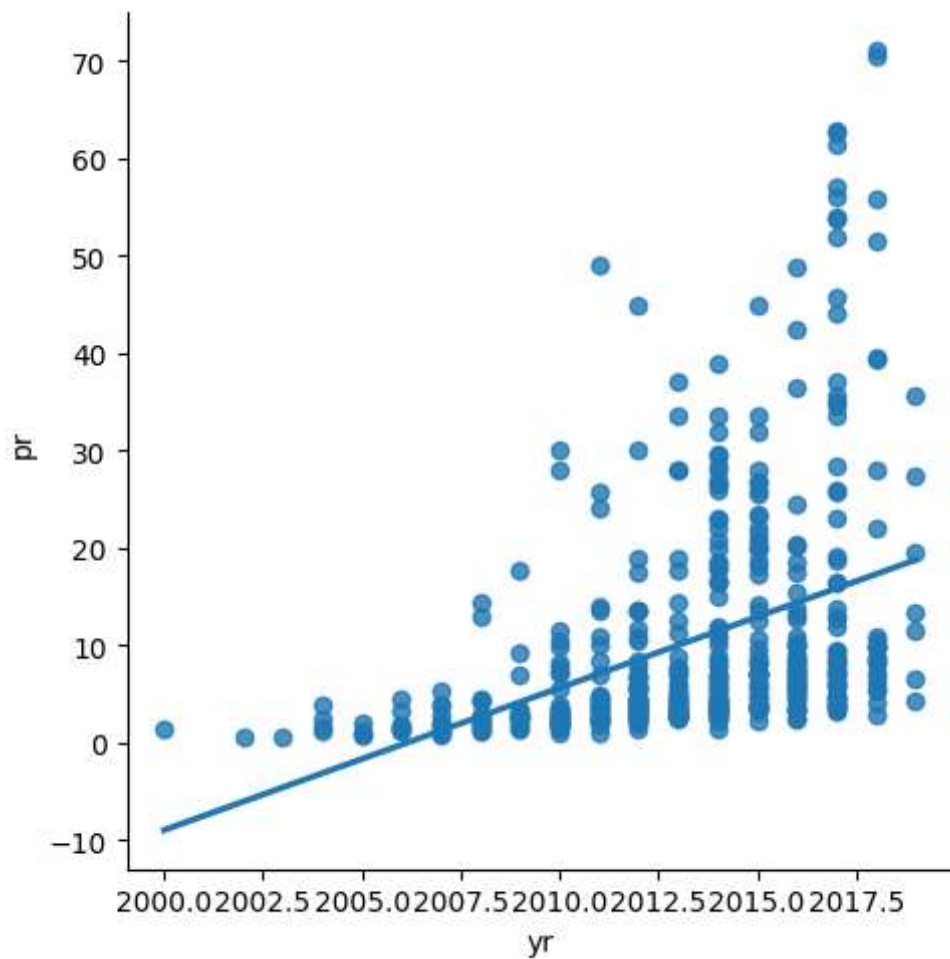


In [20]:

```
1 df500=df[:][:500]  
2 sns.lmplot(x="yr",y="pr",data=df500,order=1,ci=None)
```

Out[20]:

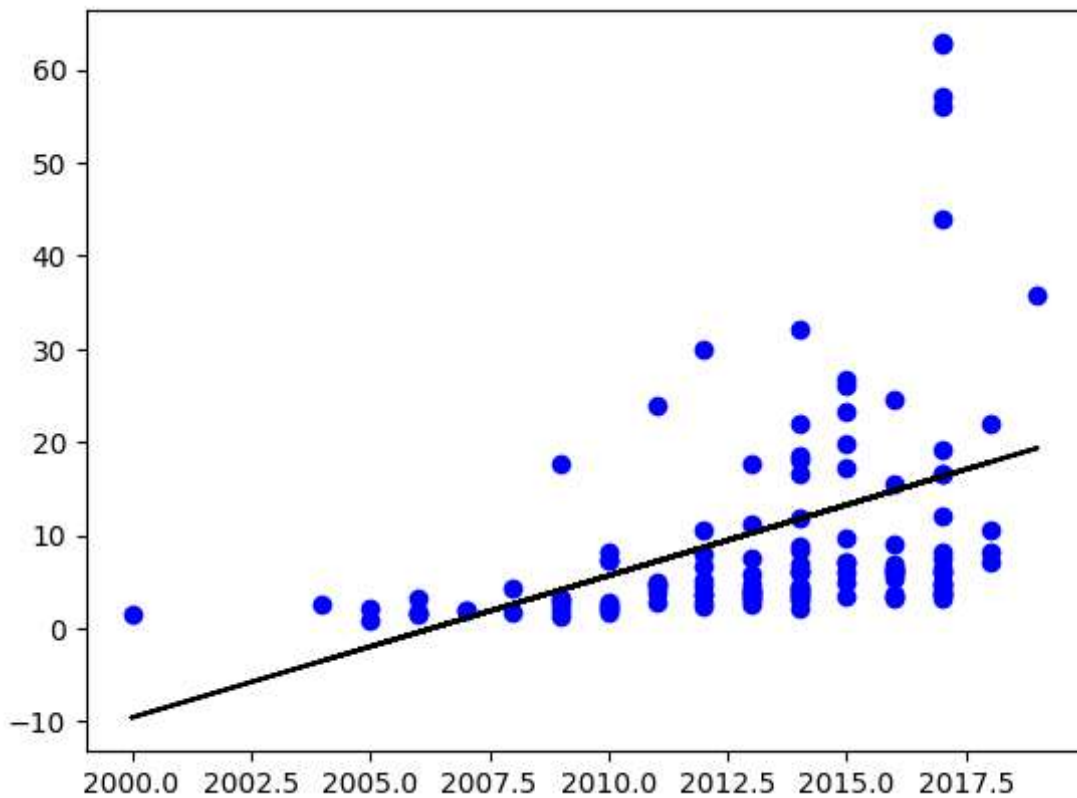
<seaborn.axisgrid.FacetGrid at 0x20fe809d650>



In [21]:

```
1 df500.fillna(method='ffill',inplace=True)
2 X=np.array(df500['yr']).reshape(-1,1)
3 y=np.array(df500['pr']).reshape(-1,1)
4 df500.dropna(inplace=True)
5 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
6 regr=LinearRegression()
7 regr.fit(X_train,y_train)
8 print("Regression:",regr.score(X_test,y_test))
9 y_pred=regr.predict(X_test)
10 plt.scatter(X_test,y_test,color='b')
11 plt.plot(X_test,y_pred,color='k')
12 plt.show()
```

Regression: 0.13616017792518997



In [22]:

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.metrics import r2_score
3 model=LinearRegression()
4 model.fit(X_train,y_train)
5 y_pred=model.predict(X_test)
6 r2=r2_score(y_test,y_pred)
7 print("R2 score:",r2)
```

R2 score: 0.13616017792518997

In []:

1

