In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import r2_score, mean_absolute_error, mean_absolute_percentage_error
```

In [2]:
```
car_data = pd.read_csv('car_sales_data.csv', header = 0, sep=',')
car_data.head()
```

Out[2]:

	Manufacturer	Model	Engine size	Fuel type	Year of manufacture	Mileage	Price
0	Ford	Fiesta	1.0	Petrol	2002	127300	3074
1	Porsche	718 Cayman	4.0	Petrol	2016	57850	49704
2	Ford	Mondeo	1.6	Diesel	2014	39190	24072
3	Toyota	RAV4	1.8	Hybrid	1988	210814	1705
4	VW	Polo	1.0	Petrol	2006	127869	4101

In [3]:
```
car_data.shape
```

Out[3]:
```
(50000, 7)
```

In [4]:
```
car_data.isnull().sum()
```

Out[4]:
```
Manufacturer         0
Model                0
Engine size          0
Fuel type            0
Year of manufacture  0
Mileage              0
Price                0
dtype: int64
```

In [5]:
```
car_data.loc[car_data.duplicated()]
```

Out[5]:

	Manufacturer	Model	Engine size	Fuel type	Year of manufacture	Mileage	Price
5426	VW	Polo	1.2	Petrol	2003	10000	8024
9862	Ford	Mondeo	1.4	Diesel	1987	224569	883
14745	BMW	Z4	2.4	Petrol	1999	12000	13410
19020	Toyota	Yaris	1.0	Petrol	1996	13500	5087
19337	VW	Polo	1.0	Petrol	2000	11500	5950
23927	VW	Polo	1.2	Petrol	2021	1000	27901
25368	VW	Golf	1.2	Diesel	2011	6000	17401
28576	VW	Polo	1.2	Petrol	2003	10000	8024
34246	VW	Passat	2.0	Diesel	2003	10000	16087
35647	Ford	Focus	1.6	Petrol	2019	2000	39636
41536	VW	Passat	1.8	Diesel	1996	13500	9394
45904	Ford	Fiesta	1.2	Petrol	2003	124092	3691

In [6]:
```
car_data = car_data.drop_duplicates()
```

In [7]:
```
car_data.duplicated().sum()
```

Out[7]:
```
np.int64(0)
```

In [8]:
```
car_data.shape
```

Out[8]: (49988, 7)

In [9]:
```
car_data.head()
```

Out[9]:

	Manufacturer	Model	Engine size	Fuel type	Year of manufacture	Mileage	Price
0	Ford	Fiesta	1.0	Petrol	2002	127300	3074
1	Porsche	718 Cayman	4.0	Petrol	2016	57850	49704
2	Ford	Mondeo	1.6	Diesel	2014	39190	24072
3	Toyota	RAV4	1.8	Hybrid	1988	210814	1705
4	VW	Polo	1.0	Petrol	2006	127869	4101

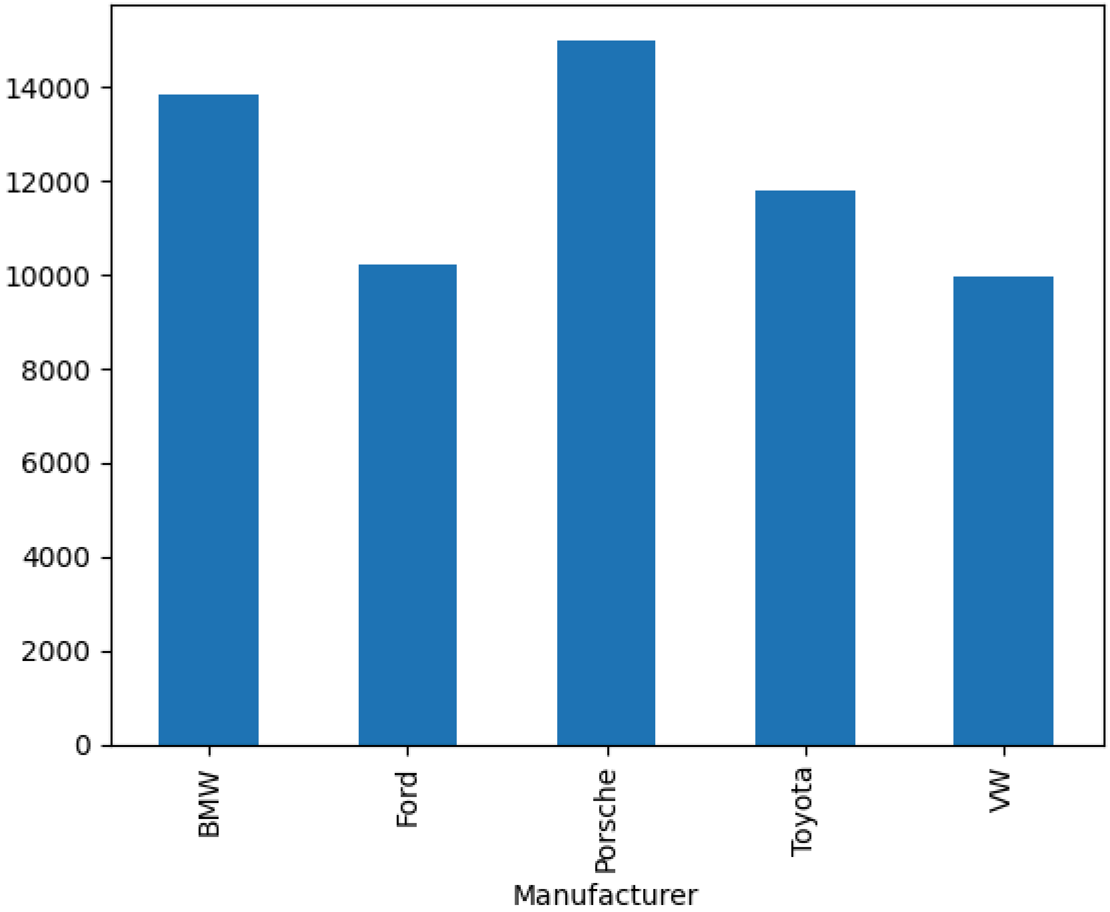
In [10]:
```
car_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 49988 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Manufacturer         49988 non-null  object 
 1   Model                49988 non-null  object 
 2   Engine size          49988 non-null  float64
 3   Fuel type            49988 non-null  object 
 4   Year of manufacture  49988 non-null  int64  
 5   Mileage              49988 non-null  int64  
 6   Price                49988 non-null  int64  
dtypes: float64(1), int64(3), object(3)
memory usage: 3.1+ MB
```

In [11]:
```
Q1 = car_data['Price'].quantile(0.25)
Q3 = car_data['Price'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR

car_data = car_data[(car_data['Price']>= lower_bound) & (car_data['Price'] <= upper_bound)]
```

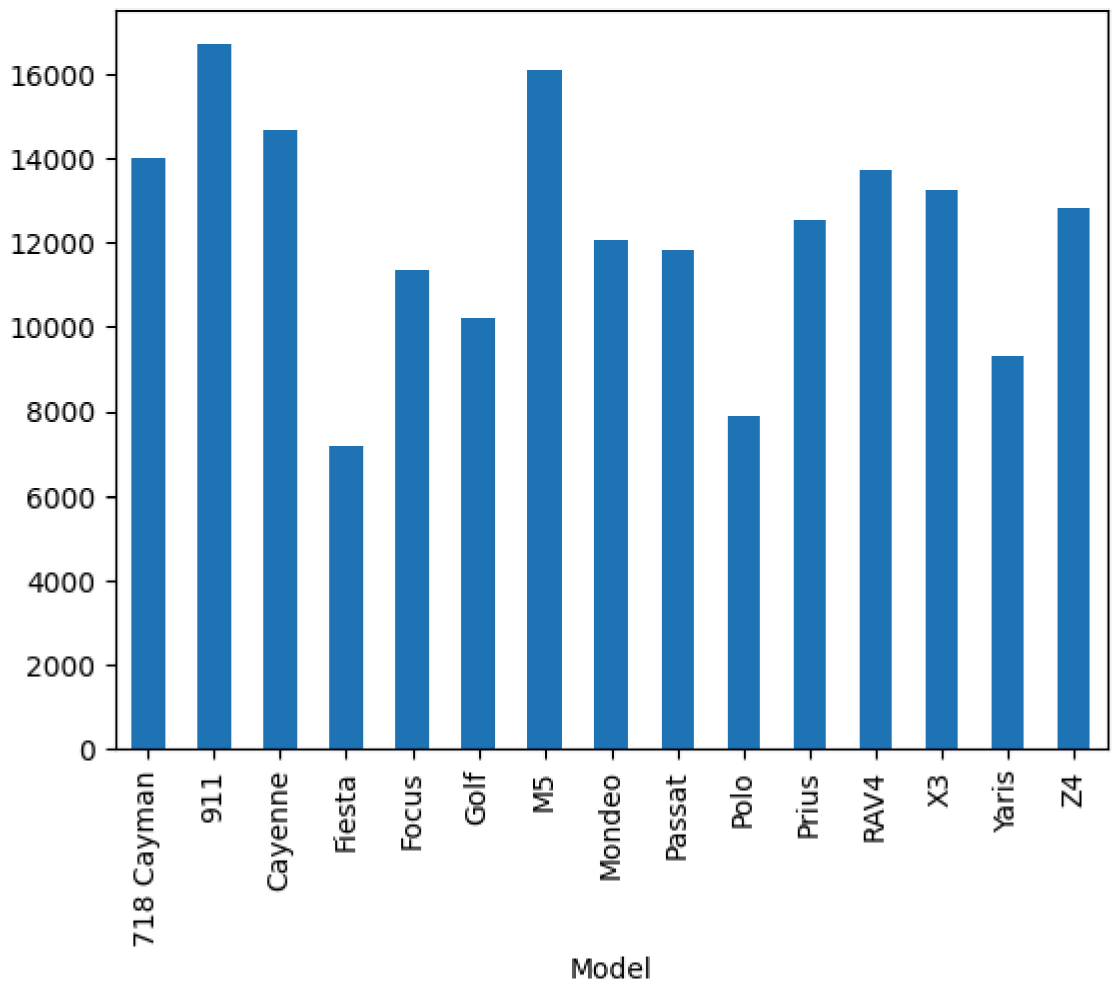
In [12]:
```
car_data.groupby('Manufacturer')['Price'].mean().plot(kind = 'bar')
```

Out[12]: <Axes: xlabel='Manufacturer'>



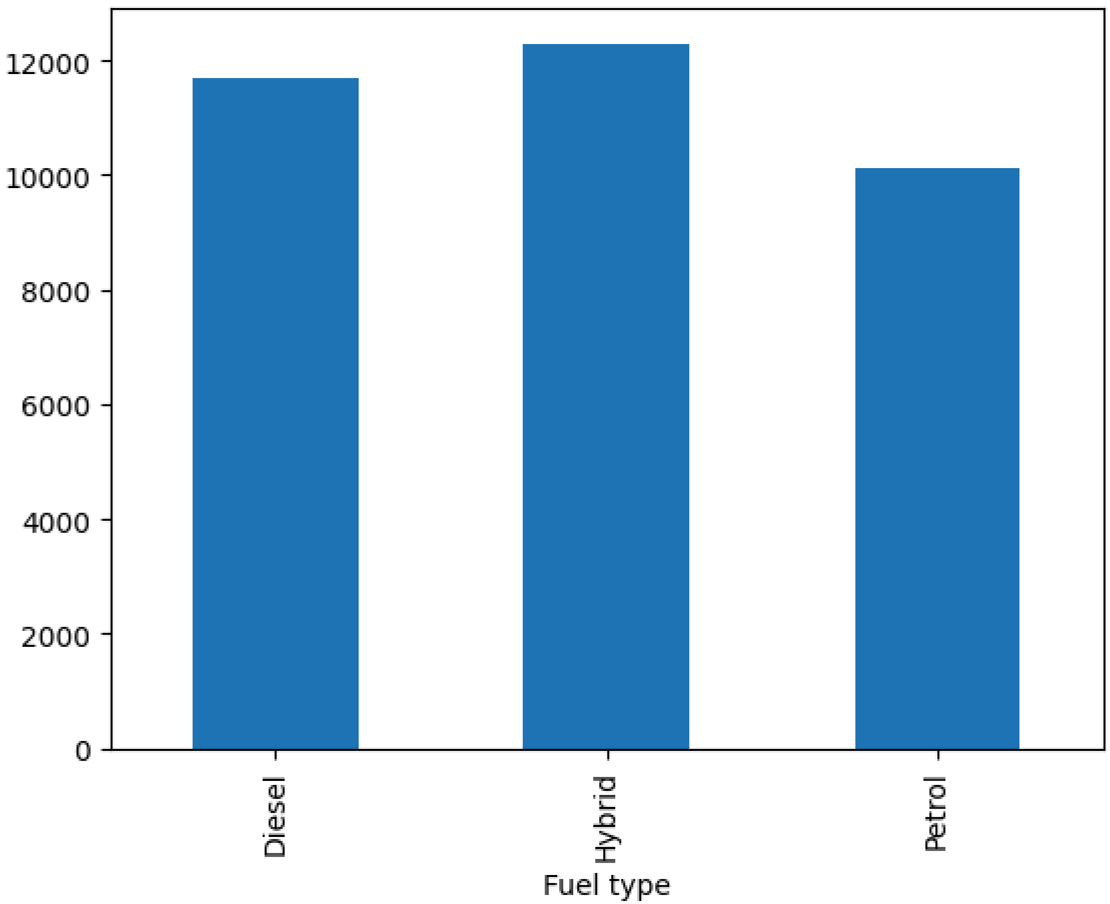
In [13]:
```
car_data.groupby('Model')['Price'].mean().plot(kind = 'bar')
```

Out[13]: <Axes: xlabel='Model'>



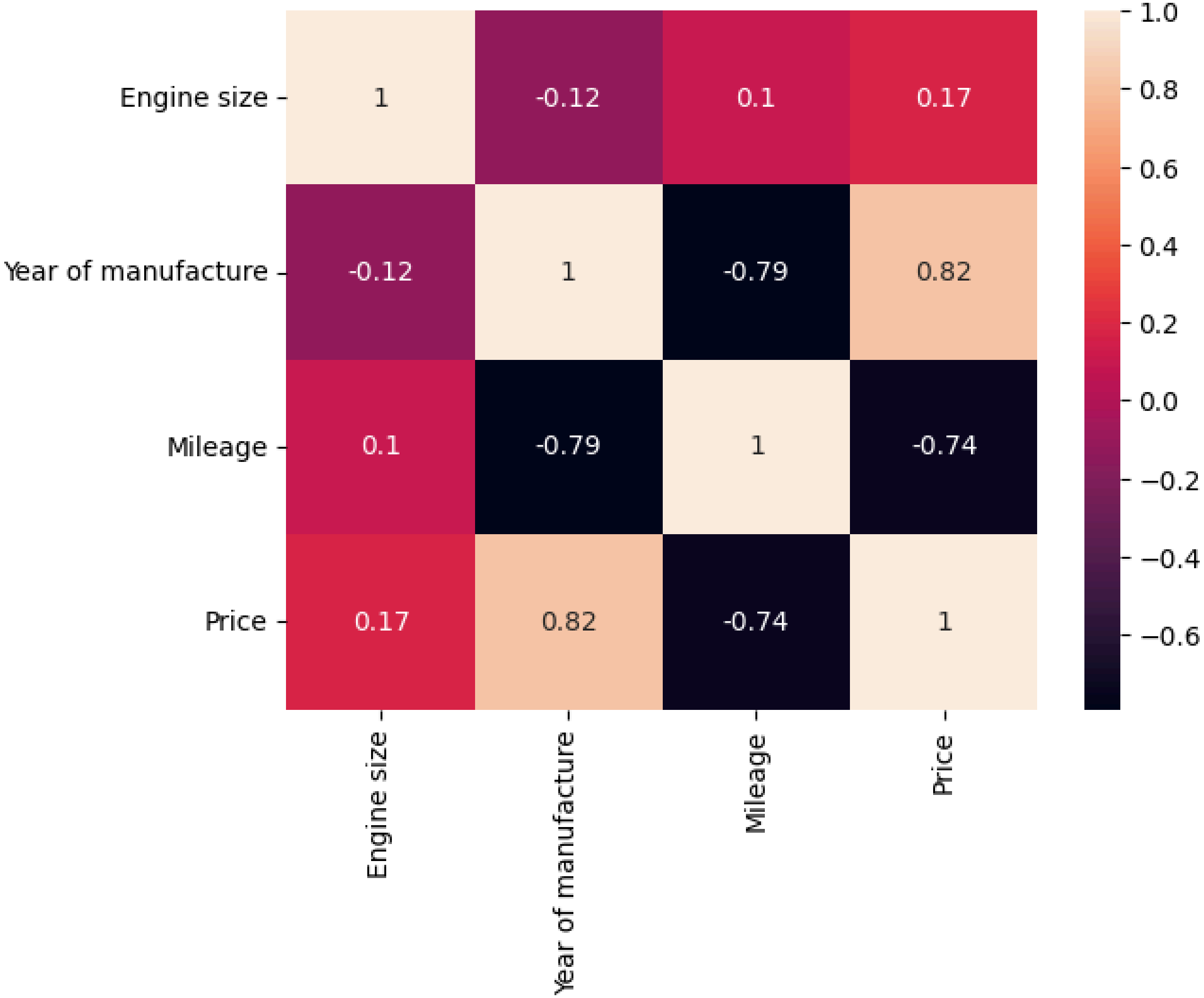
In [14]:
```
car_data.groupby('Fuel type')['Price'].mean().plot(kind = 'bar')
```

Out[14]:
```
<Axes: xlabel='Fuel type'>
```



In [15]:
```
corr_matrix = car_data.corr(numeric_only=True)

sns.heatmap(data = corr_matrix, annot = True)
```

Out[15]: <Axes: >



In [16]:
```
car_data.head()
```

Out[16]:

	Manufacturer	Model	Engine size	Fuel type	Year of manufacture	Mileage	Price
0	Ford	Fiesta	1.0	Petrol	2002	127300	3074
2	Ford	Mondeo	1.6	Diesel	2014	39190	24072
3	Toyota	RAV4	1.8	Hybrid	1988	210814	1705
4	VW	Polo	1.0	Petrol	2006	127869	4101
5	Ford	Focus	1.4	Petrol	2018	33603	29204

In [17]:

```
manufacturer = pd.get_dummies(car_data['Manufacturer'], drop_first=True, dtype = int)
manufacturer
```

Out[17]:

	Ford	Porsche	Toyota	VW
0	1	0	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
...	...	...	...	...
49993	1	0	0	0
49994	0	0	1	0
49996	0	0	1	0
49998	1	0	0	0
49999	0	0	0	1

47339 rows × 4 columns

In [18]:

```
model = pd.get_dummies(car_data['Model'], drop_first=True, dtype = int)
model
```

Out[18]:

	911	Cayenne	Fiesta	Focus	Golf	M5	Mondeo	Passat	Polo	Prius	RAV4	X3	Yaris	Z4
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49993	0	0	0	0	0	0	1	0	0	0	0	0	0	0
49994	0	0	0	0	0	0	0	0	0	0	1	0	0	0
49996	0	0	0	0	0	0	0	0	0	1	0	0	0	0
49998	0	0	0	1	0	0	0	0	0	0	0	0	0	0
49999	0	0	0	0	1	0	0	0	0	0	0	0	0	0

47339 rows × 14 columns

In [19]:

```
fuel_type = pd.get_dummies(car_data['Fuel type'], drop_first=True, dtype = int)
fuel_type
```

Out[19]:

	Hybrid	Petrol
0	0	1
2	0	0
3	1	0
4	0	1
5	0	1
...	...	...
49993	0	1
49994	1	0
49996	1	0
49998	0	0
49999	0	0

47339 rows × 2 columns

In [20]:

```
car_data = pd.concat([car_data, manufacturer, model, fuel_type], axis = 1)
car_data
```

Out[20]:

	Manufacturer	Model	Engine size	Fuel type	Year of manufacture	Mileage	Price	Ford	Porsche	Toyota	...	Mondeo	Passat	Polo	Prius	F
0	Ford	Fiesta	1.0	Petrol	2002	127300	3074	1	0	0	...	0	0	0	0	
2	Ford	Mondeo	1.6	Diesel	2014	39190	24072	1	0	0	...	1	0	0	0	
3	Toyota	RAV4	1.8	Hybrid	1988	210814	1705	0	0	1	...	0	0	0	0	
4	VW	Polo	1.0	Petrol	2006	127869	4101	0	0	0	...	0	0	1	0	
5	Ford	Focus	1.4	Petrol	2018	33603	29204	1	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49993	Ford	Mondeo	1.8	Petrol	2003	120969	6654	1	0	0	...	1	0	0	0	
49994	Toyota	RAV4	1.8	Hybrid	2002	101634	10639	0	0	1	...	0	0	0	0	
49996	Toyota	Prius	1.8	Hybrid	2003	105120	9430	0	0	1	...	0	0	0	1	
49998	Ford	Focus	1.0	Diesel	2016	26468	23630	1	0	0	...	0	0	0	0	
49999	VW	Golf	1.4	Diesel	2012	109300	10400	0	0	0	...	0	0	0	0	

47339 rows × 27 columns

In [21]:

```
car_data = car_data.drop(['Manufacturer','Model','Fuel type'], axis = 1)
car_data
```

Out[21]:

	Engine size	Year of manufacture	Mileage	Price	Ford	Porsche	Toyota	VW	911	Cayenne	...	Mondeo	Passat	Polo	Prius	RAV4	X3
0	1.0	2002	127300	3074	1	0	0	0	0	0	...	0	0	0	0	0	0
2	1.6	2014	39190	24072	1	0	0	0	0	0	...	1	0	0	0	0	0
3	1.8	1988	210814	1705	0	0	1	0	0	0	...	0	0	0	0	1	0
4	1.0	2006	127869	4101	0	0	0	1	0	0	...	0	0	1	0	0	0
5	1.4	2018	33603	29204	1	0	0	0	0	0	...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49993	1.8	2003	120969	6654	1	0	0	0	0	0	...	1	0	0	0	0	0
49994	1.8	2002	101634	10639	0	0	1	0	0	0	...	0	0	0	0	1	0
49996	1.8	2003	105120	9430	0	0	1	0	0	0	...	0	0	0	1	0	0
49998	1.0	2016	26468	23630	1	0	0	0	0	0	...	0	0	0	0	0	0
49999	1.4	2012	109300	10400	0	0	0	1	0	0	...	0	0	0	0	0	0

47339 rows × 24 columns

In [22]:

```
y = car_data['Price']
X = car_data.drop('Price', axis = 1)
```

In [23]:

```
X.head()
```

Out[23]:

	Engine size	Year of manufacture	Mileage	Ford	Porsche	Toyota	VW	911	Cayenne	Fiesta	...	Mondeo	Passat	Polo	Prius	RAV4	X3	Yaris
0	1.0	2002	127300	1	0	0	0	0	0	1	...	0	0	0	0	0	0	0
2	1.6	2014	39190	1	0	0	0	0	0	0	...	1	0	0	0	0	0	0
3	1.8	1988	210814	0	0	1	0	0	0	0	...	0	0	0	0	1	0	0
4	1.0	2006	127869	0	0	0	1	0	0	0	...	0	0	1	0	0	0	0
5	1.4	2018	33603	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 23 columns

Using Linear Regression Model.

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)

price_model = LinearRegression()
price_model.fit(X_train, y_train)
pre_y = price_model.predict(X_test)
```

In [25]:
```
r2_score = r2_score(y_test, pre_y)
r2_score
```

Out[25]: 0.808185698127409

In [26]:
```
mae = mean_absolute_error(y_test, pre_y)
mae
```

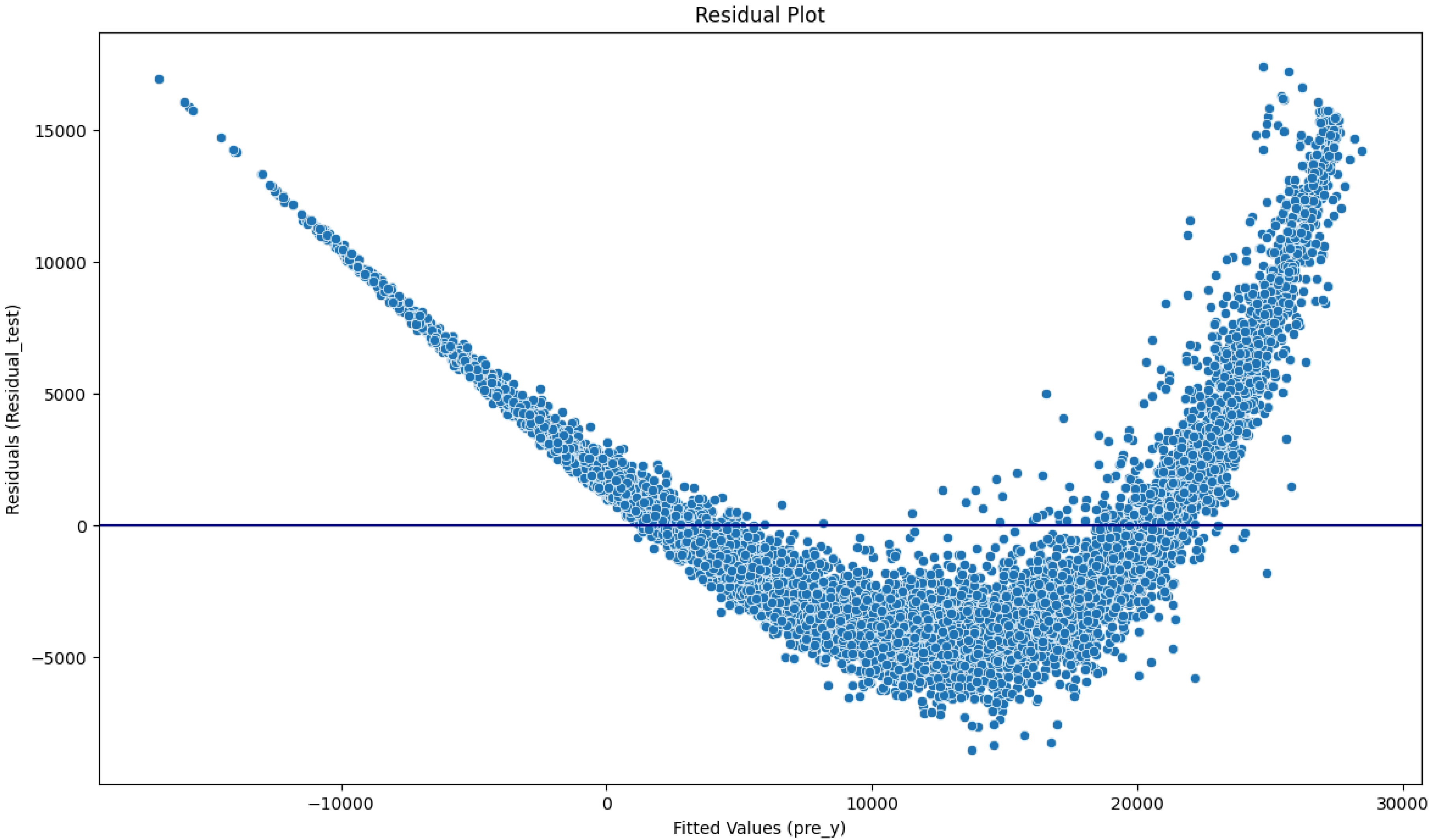
Out[26]: 3561.330915345473

In [27]:
```
residual_test = y_test - pre_y

fig = px.scatter(x = pre_y, y = residual_test, title = 'Residual Plot')
fig.add_hline(y = 0, line_color = 'darkblue')
fig.update_layout(xaxis_title = 'Fitted Values(pre_y)', yaxis_title = 'Residual (Residual_test)')
fig.show()
```

In [28]:
```
plt.figure(figsize = (14,8))
sns.scatterplot(x = pre_y, y =residual_test)
```

```python
plt.axhline(y = 0, color = 'darkblue')
plt.title('Residual Plot')
plt.xlabel('Fitted Values (pre_y)')
plt.ylabel('Residuals (Residual_test)')
```

Out[28]: `Text(0, 0.5, 'Residuals (Residual_test)')`



As per the above calculations, R Squared is closer to 1. That means, regression relation is strong. In other words, the feature variables are largely contributes to the price of a car. But here, the Mean Absolute Error is higher. We can see by looking at the Residual plot. It having a pattern between y_test and pre_y. It indicates eventhough it had higher R Squared value, Linear Regression model is not suitable here. Since, the relationship is not linear, the models such as Ridge, Lasso and ElasticNet also not suit here.

Using Random Forest Regressor

In [35]:
```
model2 = RandomForestRegressor()

model2.fit(X_train,y_train)
pre_y2 = model2.predict(X_test)
```

In [30]:
```
mae2 = mean_absolute_error(y_test,pre_y2)
mae2
```

Out[30]: 246.15752217997465

In [31]:
```
mape2 = mean_absolute_percentage_error(y_test,pre_y2)*100
mape2
```

Out[31]: 3.0332078100578395

Random Forest Regressor model is biased to overfitting. That means it can be perfect for the train data set, but not for the test data set. Then it can give wrong predictions which can lead to desaster. Therefore I have checked the Mean Absolute Error for train data set as well. it indicates there is no huge difference between train data predictions and test data predictions.

In [32]:
```
pred_y_train = model2.predict(X_train)
mae2_1 = mean_absolute_error(y_train, pred_y_train)
mae2_1
```

Out[32]: 92.27049193314144

We can also use cross validation score for this.

In [ ]:
```
scores = cross_val_score(model2, X, y, scoring = "neg_mean_absolute_error", cv=5)
print("CV MAE: ",-np.mean(scores))
```

```
CV MAE:  241.93655576304485
```

In here, the Train MAE is less than the Test MAE significantly. Therefore there is a overfitting exists. That means, eventhough the RandomForestRegressor model is best fit for Train dataset, it can make significant error in pricing for new data. Therefore, RandomForestRegressor model is not suitable for this data set.

Using XGBRegressor

In [ ]:

```
model3 = XGBRegressor()
model3.fit(X_train, y_train)
pre_y3 = model3.predict(X_test)
```

In [ ]:

```
mae3 = mean_absolute_error(y_test, pre_y3)
mae3
```

Out[ ]:

```
291.56622314453125
```

Cat Boost Regressor

In [52]:

```
model4 = CatBoostRegressor(verbose=0)
model4.fit(X_train, y_train)
pre_y4 = model4.predict(X_test)
```

In [ ]:

```
mae4 = mean_absolute_error(y_test, pre_y4)
mae4
```

Out[ ]:

```
114.24390050325427
```

In [ ]:

```
pred_y_train4 = model4.predict(X_train)

mae4_train = mean_absolute_error(y_train, pred_y_train4 )
mae4_train
```

Out[ ]:

```
103.7946137725905
```

In [53]:

```
score4 = cross_val_score(model4, X, y, scoring = "neg_mean_absolute_error", cv = 5, verbose=0)

mae_scores = -score4
mean_mae = mae_scores.mean()
std_mae  = mae_scores.std()
print("Mean MAE: ",round(mean_mae,3))
print("STD MAE: ",round(std_mae,3))
```

```
Mean MAE:  111.769
STD MAE:  1.706
```

This shows CV MAE is closer to the test data MAE. Therefore we can conclude there is no overfitting at all. This can be confirmes by looking at the cross validation scores.

Using Decision Tree Regressor Model

In [ ]:
```
model5 = DecisionTreeRegressor()

model5.fit(X_train,y_train)
pre_y5 = model5.predict(X_test)
```

In [ ]:
```
mae5 = mean_absolute_error(y_test,pre_y5)
mae5
```

Out[ ]: 400.06738487536967

In [ ]:
```
mape5 = mean_absolute_percentage_error(y_test, pre_y5)*100
mape5
```

Out[ ]: 5.043911029984973

Gradient Boosting regressor

In [47]:
```
model6 = GradientBoostingRegressor()
model6.fit(X_train, y_train)
pre_y6 = model6.predict(X_test)
```

In [48]:
```
mae6 = mean_absolute_error(y_test, pre_y6)
mae6
```

Out[48]: 851.1376058007393

In [50]:
```
model7 = MLPRegressor(max_iter = 500)
model7.fit(X_train, y_train)
pre_y7 = model7.predict(X_test)
```

In [51]:
```
mae7 = mean_absolute_error(y_test, pre_y7)
mae7
```

Out[51]: 2898.446006154152

In Conclution, Cat Boost Regressor is the best model that fit to this data set. It has lower and closer Train and Test Mean Absolute Error, and also low mean MAE and low standard deviated MAE.