In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
```

In [2]:
```
#importing data
df=pd.read_csv("C:\\Users\\MMU\\Desktop\\used-cars.csv")
df
```

Out[2]:

	Transmission	kilometres driven	Price
0	Manual	72000	1.75
1	Manual	41000	12.50
2	Manual	46000	4.50
3	Manual	87000	6.00
4	Automatic	40670	17.74
...	...	...	...
94	Manual	58000	8.10
95	Manual	30000	3.50
96	Manual	34212	2.79
97	Manual	70002	3.45
98	Manual	62000	4.45

99 rows × 3 columns

In [30]:
```
#calling arrays
```

In [31]:
```
x= np.array(df["kilometres driven"]).reshape(-1,1)
```

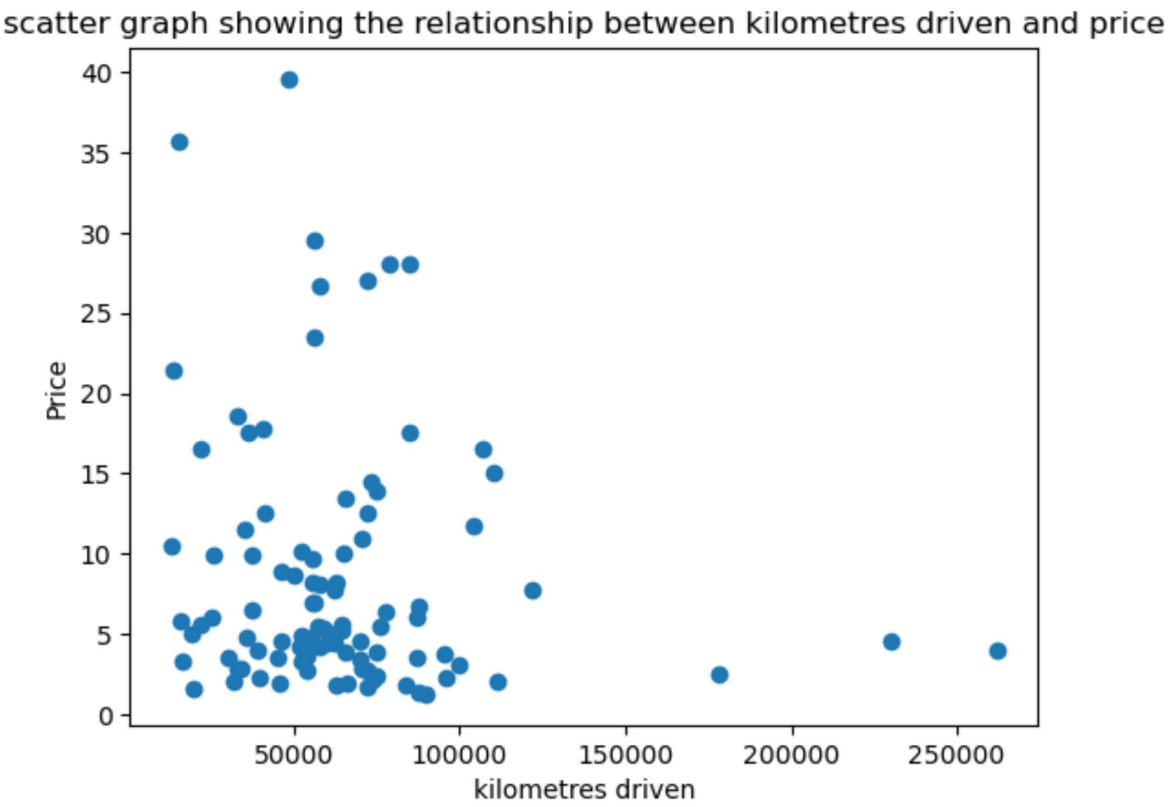
In [32]:
```
y= np.array(df["Price"])
```

In [33]:
```
#checking for missing data
df.isna().sum()
```

Out[33]:
```
Transmission         0
kilometres driven    0
Price                0
dtype: int64
```

In [34]:
```
#visualization of the graph
```

In [36]:
```
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.xlabel("kilometres driven")
plt.ylabel("Price")
plt.title("scatter graph showing the relationship between kilometres driven and pri
#plt.grid(True)
plt.show()
```



In [38]:
```
#spliting data
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)
```

In [40]:
```
#standardising data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled =scaler.transform(x_test)
```

In [43]:
```
#building model
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

In [44]:
```
#model fitting
model.fit(x_train_scaled,y_train)
```

Out[44]:
```
▾ LinearRegression

LinearRegression()
```

In [46]:
```
#making prediction
y_pred=model.predict(x_test_scaled)
y_pred
```

Out[46]:
```
array([8.24742366, 8.77694736, 9.24376033, 9.79443976, 8.82737819,
       7.06703969, 8.51410188, 8.72485232, 8.51395059, 8.36500314,
       8.82737819, 9.58300851, 8.72651653, 8.24742366, 8.4995778 ,
       9.50819438, 8.67595963, 8.96924011, 9.14761395, 9.11306883])
```

In [47]:
```
#getting coeefficient
model.coef_
```

Out[47]:
```
array([-1.00112988])
```

In [48]:
```
#getting intercept
model.intercept_
```

Out[48]:
```
8.477468354430382
```

In [50]:
```
#model accuracy on train values
model.score(x_train_scaled,y_train)
```

Out[50]:
```
0.015326935308310308
```

In [51]:
```
#model accuracy on test values
model.score(x_test_scaled,y_test)
```

Out[51]:
```
0.016274635870596632
```

In [55]:
```
from sklearn.metrics import mean_absolute_error,r2_score,mean_squared_error
mae=mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)

print(f"mae:{mae}")
print(f"r2:{r2}")
print(f"mse:{mse}")
```

```
mae:5.245837297661162
r2:0.016274635870596632
mse:46.88144886458319
```

MODEL OPTIMIZATION

In [63]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge

#perform GridsearchCV to find optimal alpha for ridge Regression
param_grid={"alpha":[0.1,1,10,100]}
ridge_model=Ridge()
grid_search = GridSearchCV(ridge_model,param_grid,cv=5)
grid_search.fit(x_train_scaled,y_train)
best_alpha = grid_search.best_params_["alpha"]

#Train ridge Regression model with the best alpha
ridge_model = Ridge(alpha=best_alpha)
ridge_model.fit(x_train_scaled,y_train)

#make prediction
y_pred_ridge=ridge_model.predict(x_test_scaled)

#evaluate model prformance
mae_ridge=mean_absolute_error(y_test,y_pred_ridge)
r2_ridge=r2_score(y_test,y_pred_ridge)
mse_ridge=mean_squared_error(y_test,y_pred_ridge)

print("Ridge Regression")
print(f"Best alpha: {best_alpha}")
print(f"MAE:{mae_ridge}")
print(f"R^2:{r2_ridge}")
print(f"MSE:{mse_ridge}")
```

```
Ridge Regression
Best alpha: 100
MAE:5.141801661150186
R^2:0.00927875460315164
MSE:47.214851927939876
```

In [ ]: