In [133]:

```
import pandas as pd
import csv
```

In [134]:

```
fuel = pd.read_csv('fuel_data.csv')
fuel
```

Out[134]:

	drivenKM	fuelAmount
0	390.00	3600.0
1	403.00	3705.0
2	396.50	3471.0
3	383.50	3250.5
4	321.10	3263.7
5	391.30	3445.2
6	386.10	3679.0
7	371.80	3744.5
8	404.30	3809.0
9	392.20	3905.0
10	386.43	3874.0
11	395.20	3910.0
12	381.00	4020.7
13	372.00	3622.0
14	397.00	3450.5
15	407.00	4179.0
16	372.40	3454.2
17	375.60	3883.8
18	399.00	4235.9

In [135]:

```
fuel.head()
```

Out[135]:

	drivenKM	fuelAmount
0	390.0	3600.0
1	403.0	3705.0
2	396.5	3471.0
3	383.5	3250.5
4	321.1	3263.7

In [136]:

```
fuel.shape
```

Out[136]:

```
(19, 2)
```

In [137]:

```
fuel.columns
```

Out[137]:

```
Index(['drivenKM', 'fuelAmount'], dtype='object')
```

In [138]:

```
fuel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19 entries, 0 to 18
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   drivenKM    19 non-null     float64
 1   fuelAmount  19 non-null     float64
dtypes: float64(2)
memory usage: 432.0 bytes
```

In [139]:

```
fuel.isnull()
```

Out[139]:

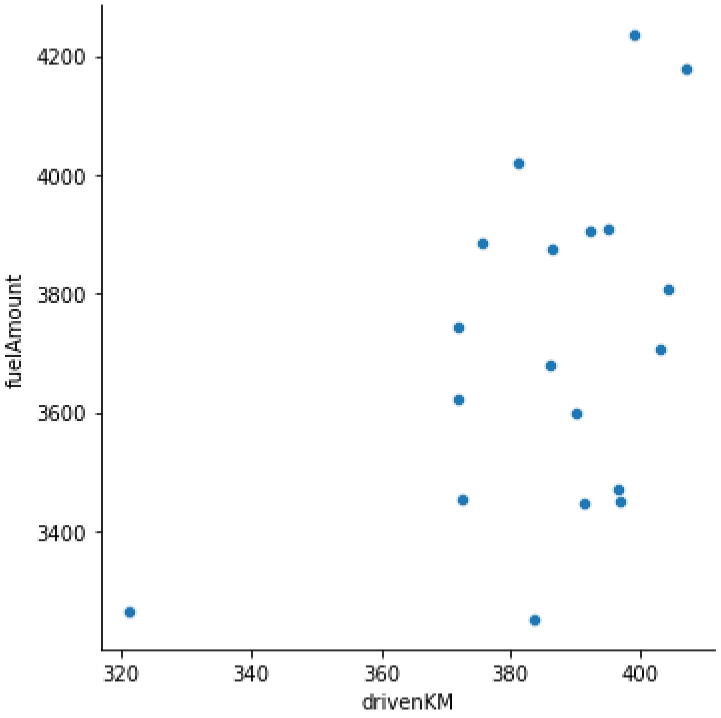
	drivenKM	**fuelAmount**
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False

In [140]:

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

```
In [141]: sns.relplot(x="drivenKM", y="fuelAmount",data=fuel)

Out[141]: <seaborn.axisgrid.FacetGrid at 0x23b5cb31550>
```



```
In [142]: data1 = ['drivenKM']
          X=fuel[data1]
```

```
In [143]: data2 = ['fuelAmount']
          Y=fuel[data2]
```

In [144]:
```
print(X)
X.dtypes
```

```
    drivenKM
0     390.00
1     403.00
2     396.50
3     383.50
4     321.10
5     391.30
6     386.10
7     371.80
8     404.30
9     392.20
10    386.43
11    395.20
12    381.00
13    372.00
14    397.00
15    407.00
16    372.40
17    375.60
18    399.00
```

Out[144]:
```
drivenKM    float64
dtype: object
```

In [145]:
```
print(Y)
Y.dtypes
y_True=y
```

```
    fuelAmount
0       3600.0
1       3705.0
2       3471.0
3       3250.5
4       3263.7
5       3445.2
6       3679.0
7       3744.5
8       3809.0
9       3905.0
10      3874.0
11      3910.0
12      4020.7
13      3622.0
14      3450.5
15      4179.0
16      3454.2
17      3883.8
18      4235.9
```

In [146]:
```
from sklearn.model_selection import train_test_split
```

In [147]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.8,test_size=
```

In [148]:
```
X_train, X_test, Y_train, Y_test
```

Out[148]:
```
(    drivenKM
 9     392.20
 8     404.30
 5     391.30
 1     403.00
 14    397.00
 4     321.10
 16    372.40
 6     386.10
 7     371.80
 15    407.00
 10    386.43
 13    372.00
 17    375.60
 18    399.00
 3     383.50,
     drivenKM
 11     395.2
 12     381.0
 0      390.0
 2      396.5,
     fuelAmount
 9       3905.0
 8       3809.0
 5       3445.2
 1       3705.0
 14      3450.5
 4       3263.7
 16      3454.2
 6       3679.0
 7       3744.5
 15      4179.0
 10      3874.0
 13      3622.0
 17      3883.8
 18      4235.9
 3       3250.5,
     fuelAmount
 11      3910.0
 12      4020.7
 0       3600.0
 2       3471.0)
```

In [149]:
```
X_train.shape
```

Out[149]:
```
(15, 1)
```

In [150]:
```
X_test.shape
```

Out[150]:
```
(4, 1)
```

```
In [151]: Y_train.shape

Out[151]: (15, 1)

In [152]: Y_test.shape

Out[152]: (4, 1)

In [153]: from sklearn.linear_model import LinearRegression

In [154]: lin_reg = LinearRegression()
          lin_reg.fit(X_train,Y_train)

Out[154]: LinearRegression()

In [155]: x=[[800]]
          lin_reg.predict(x)

Out[155]: array([[6857.09812249]])

In [156]: y_pred=lin_reg.predict(X)
          y_pred

Out[156]: array([[3744.25861846],
                 [3842.95840761],
                 [3793.60851303],
                 [3694.90872388],
                 [3221.14973595],
                 [3754.12859737],
                 [3714.64868171],
                 [3606.07891365],
                 [3852.82838653],
                 [3760.9616597 ],
                 [3717.1541379 ],
                 [3783.73853412],
                 [3675.9279952 ],
                 [3607.59737194],
                 [3797.40465877],
                 [3873.3275735 ],
                 [3610.63428853],
                 [3634.92962124],
                 [3812.58924172]])

In [157]: from sklearn.metrics import mean_squared_error, r2_score


          mse = mean_squared_error(y_True, y_pred)
          mse

Out[157]: 58987.44292405615

In [158]: r2_score(y_True, y_pred)

Out[158]: 0.21643923225776895
```

In [159]:

```
lin_reg.coef_
```

Out[159]:

```
array([[7.59229147]])
```

In [160]:

```
lin_reg.intercept_
```

Out[160]:

```
array([783.2649439])
```

In [161]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
s_X_train=scaler.fit_transform(X_train)
s_X_train
```

Out[161]:

```
array([[ 0.39182783],
       [ 0.98313698],
       [ 0.34784616],
       [ 0.91960789],
       [ 0.62639675],
       [-3.08272426],
       [-0.57576896],
       [ 0.09372983],
       [-0.60509007],
       [ 1.11508199],
       [ 0.10985644],
       [-0.59531637],
       [-0.41938968],
       [ 0.7241338 ],
       [-0.03332833]])
```

In [162]:

```
s_X_test=scaler.transform(X_test)
s_X_test
```

Out[162]:

```
array([[ 0.5384334 ],
       [-0.15549964],
       [ 0.28431708],
       [ 0.60196248]])
```

In [163]:

```
s_lin_reg = LinearRegression()
s_lin_reg.fit( s_X_train,y_train)
```

Out[163]:

```
LinearRegression()
```

In [164]:

```
s_y_pred=s_lin_reg.predict(s_X_test)
s_y_pred
```

Out[164]:

```
array([[3745.68215321],
       [3676.96914641],
       [3720.51964368],
       [3751.97278059]])
```

In [165]:

```
mean_squared_error(y_test, s_y_pred)
```

Out[165]:

```
67297.78117999973
```

In [166]:
```
r2_score(y_test, s_y_pred)
```

Out[166]:
```
0.0345055936007026
```

In [167]:
```
s_lin_reg.coef_
```

Out[167]:
```
array([[99.01964899]])
```

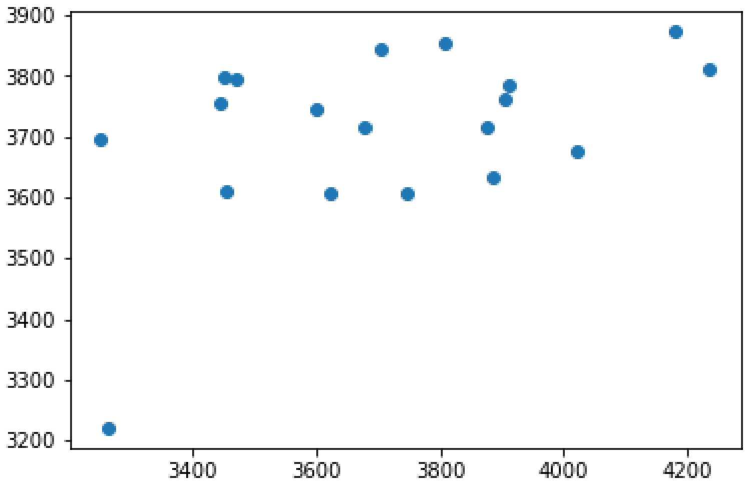
In [168]:
```
s_lin_reg.intercept_
```

Out[168]:
```
array([3692.36666667])
```

In [169]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.scatter(y_True,y_pred)
```

Out[169]:
```
<matplotlib.collections.PathCollection at 0x23b5cf242b0>
```



In [170]:
```
from sklearn.preprocessing import MinMaxScaler
scalers = MinMaxScaler()
m_X_train=scalers.fit_transform(X_train)
m_X_train
```

Out[170]:
```
array([[0.82770664],
       [0.9685681 ],
       [0.81722934],
       [0.95343423],
       [0.88358556],
       [0.        ],
       [0.59720605],
       [0.75669383],
       [0.59022119],
       [1.        ],
       [0.76053551],
       [0.59254948],
       [0.63445867],
       [0.90686845],
       [0.72642608]])
```

```
In [171]: m_X_test=scalers.transform(X_test)
          m_X_test

Out[171]: array([[0.86263097],
                 [0.69732247],
                 [0.80209546],
                 [0.87776484]])
```

```
In [172]: m_lin_reg = LinearRegression()
          m_lin_reg.fit( m_X_train,y_train)

Out[172]: LinearRegression()
```

```
In [173]: m_y_pred=m_lin_reg.predict(m_X_test)
          m_y_pred

Out[173]: array([[3745.68215321],
                 [3676.96914641],
                 [3720.51964368],
                 [3751.97278059]])
```

```
In [174]: mean_squared_error(y_test, m_y_pred)

Out[174]: 67297.7811799998
```

```
In [175]: r2_score(y_test, m_y_pred)

Out[175]: 0.03450559360070149
```

```
In [176]: import numpy as np
          from sklearn.neighbors import KNeighborsRegressor
          neigh = KNeighborsRegressor(n_neighbors=5)
          neigh.fit(X, y)

Out[176]: KNeighborsRegressor()
```

In [177]:
```
n_y_pred=neigh.predict(X)
n_y_pred
```

Out[177]:
```
array([[3700.64],
       [3875.88],
       [3794.48],
       [3684.84],
       [3593.64],
       [3746.84],
       [3684.84],
       [3745.04],
       [3875.88],
       [3666.24],
       [3569.74],
       [3794.48],
       [3741.6 ],
       [3745.04],
       [3794.48],
       [3875.88],
       [3745.04],
       [3745.04],
       [3754.48]])
```

In [178]:
```
knn_mse=mean_squared_error(y_True, n_y_pred)
knn_mse
```

Out[178]:
```
70460.30507368421
```

In [179]:
```
r2_score(y_True,n_y_pred)
```

Out[179]:
```
0.06403925984775638
```

In [180]:
```
import numpy as np
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
# Always scale the input. The most convenient way is to use a pipeline.
reg = make_pipeline(StandardScaler(), SGDRegressor(max_iter=1000, tol=1e-3))
reg.fit(X, y)
```

```
C:\Users\VISSWES\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[180]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('sgdregressor', SGDRegressor())])
```

In [181]:

```
r_y_pred=reg.predict(X)
r_y_pred
```

Out[181]:

```
array([3740.41973845, 3830.0665893 , 3785.24316387, 3695.59631303,
       3265.29142895, 3749.38442353, 3713.52568319, 3614.91414726,
       3839.03127438, 3755.59074398, 3715.80133402, 3776.27847879,
       3678.35653402, 3616.29332958, 3788.69111968, 3857.65023571,
       3619.05169422, 3641.11861136, 3802.48294288])
```

In [182]:

```
sgd_mse=mean_squared_error(y_True,r_y_pred)
sgd_mse
```

Out[182]:

```
58824.133120863604
```

In [183]:

```
r2_score(y_True,r_y_pred)
```

Out[183]:

```
0.218608560989888
```

In [184]:

```
print('LR:',mse)
print('KNNR:',knn_mse)
print('SGDR:',sgd_mse)
```

```
LR: 58987.44292405615
KNNR: 70460.30507368421
SGDR: 58824.133120863604
```

In [ ]:

```
```

In [ ]:

```
```