GeneralModeling

March 29, 2021

1 General Regressions and Modeling

[1]:
```
%load_ext autoreload
%autoreload 2
```

[155]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[6]:
```
import os
import sys
HOME_DIR = '/home/aysola/'
os.chdir(HOME_DIR)
sys.path.append('./NVFEL498/')
```

[3]:
```
from Notebooks.utils import *
```

[9]:
```
data = pd.read_csv('./NVFEL498/alltrips_vel.csv')
```

[12]:
```
print(data.shape)
```

```
(11803, 15)
```

[11]:
```
data.head()
```

[11]:
```
   Unnamed: 0  Unnamed: 0.1  Unnamed: 0.1.1  TripId  TripId_raw  VehId  \
0           0             0               0       0        2265    123
1           1             1               1       1        1239    135
2           2             2               2       4        2263    575
3           3             3               3       7        1681    522
4           4             4               4       8        1375    250

   Aggressivity  Aggressiveness  Distance[km]  Fuel Consumed[L]  \
0    475.645508     5748.696207      9.225222          0.895646
1    269.540959     6372.966896      2.223611          0.441901
2    266.112770     5665.167276      6.323556          1.008958
3    552.585767     5891.611649      2.743222          0.344075
```

```
4    529.245807     5535.915731      4.683417          0.937469

   Fuel Rate[gpm]  Weight  Displacement  SpeedAverage  SpeedVariance
0        0.097087  2500.0           1.8     46.423729     147.494133
1        0.198731  3500.0           2.5     40.748170     447.766401
2        0.159556  4000.0           2.4     32.929095     247.070941
3        0.125427  3000.0           2.4     33.915000     447.276871
4        0.200168  5500.0           5.3     33.196768     319.067811
```

[15]:
```
data['SpeedAverage'].isna().sum()
```

[15]: 0

[16]:
```
data['SpeedVariance'].isna().sum()
```

[16]: 0

1.1 Lets build a linear regression

[17]:
```
# lets build a super simple stupid regression
from sklearn.linear_model import LinearRegression
```

[246]:
```
# Before we can fit we have to do some preprocessing
features = ['Aggressiveness', 'Distance[km]', 'Weight', 'Displacement',␣
 ↪'SpeedAverage', 'SpeedVariance']
target = ['Fuel Rate[gpm]']
```

[127]:
```
# WEIGHT HAS SOME NAN VALUES THAT ARE ENCODED AS FLOATS -- WE NEED TO USE NP.
 ↪ISNAN INSTEAD OF PANDAS
#
mask = [~(np.isnan(data.iloc[i,:]).sum()>0) for i in range(len(data))]
data = data.loc[mask]
```

[212]:
```
X = np.array(data.loc[:,features])
```

[213]:
```
y = np.array(data.loc[:,'Fuel Rate[gpm]']).reshape(-1,1)
```

[214]:
```
X.shape
```

[214]: (11529, 6)

[215]:
```
y.shape
```

[215]: (11529, 1)

[216]:
```
model = LinearRegression().fit(X,y)
```

[217]:
```
for feature, coef in zip(features, model.coef_[0]):
    print(f"{feature}: {coef}")
```

```
Aggressiveness: 1.188820516820224e-05
Distance[km]: -3.8248981102592604e-05
Weight: 6.059327624259096e-06
Displacement: 0.0299524660281594
SpeedAverage: 1.707837633178612e-05
SpeedVariance: 4.868811892222443e-08
```

[218]:
```
# lets see which value has the highest average impact on our regression:

print('Impact = MEAN * Coef')
for i, vals in enumerate(zip(features, model.coef_[0])):
    feature, coef = vals
    print(f"{feature}: {X[:,i].mean()*coef}")

# we can actually get the relative percentages here:
print('========= PERCENTAGES =========')

impact = np.array([np.abs(coef * X[:,i].mean()) for i, coef in enumerate(model.
 ↪coef_[0])])
for feature, i in zip(features, (impact*100)/impact.sum()):
    print(f"{feature}: {i}")
```

```
Impact = MEAN * Coef
Aggressiveness: 0.06184528612055701
Distance[km]: -0.00020453144069008174
Weight: 0.020393534577180465
Displacement: 0.07506926104927253
SpeedAverage: 0.0006729971659087428
SpeedVariance: 2.4989540075156042e-05
========= PERCENTAGES =========
Aggressiveness: 39.090482029722686
Distance[km]: 0.12927796293517937
Weight: 12.890118987529737
Displacement: 47.44894532965449
SpeedAverage: 0.4253805790263045
SpeedVariance: 0.015795111131585857
```

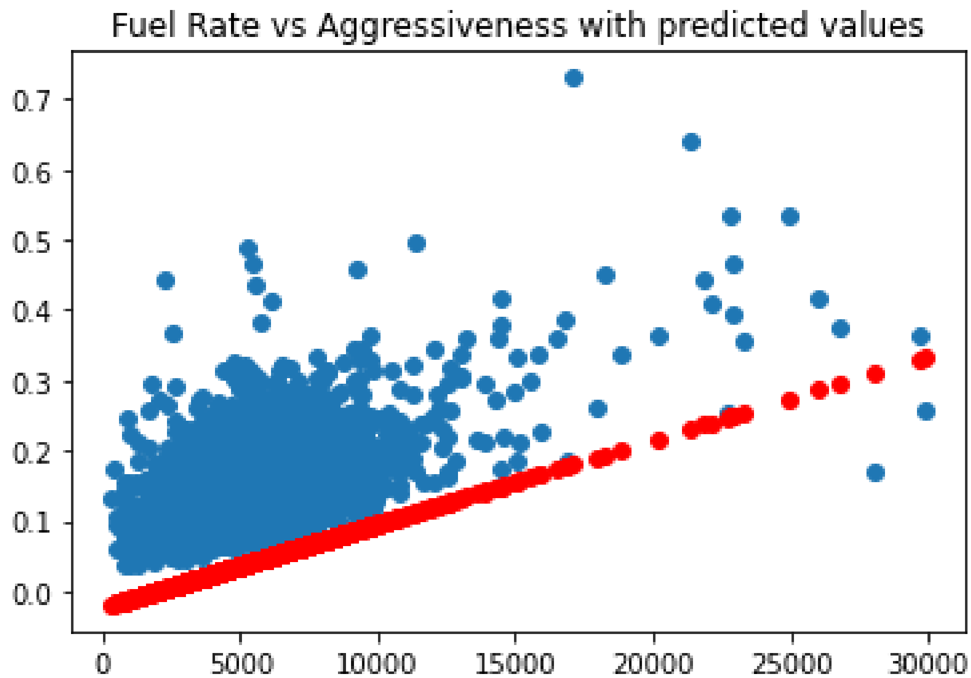
[219]:
```
# now lets compute r^2:
preds = model.predict(X)
r_2 = 1 - (((y - preds)**2).sum())/((y - y.mean())**2).sum()
```

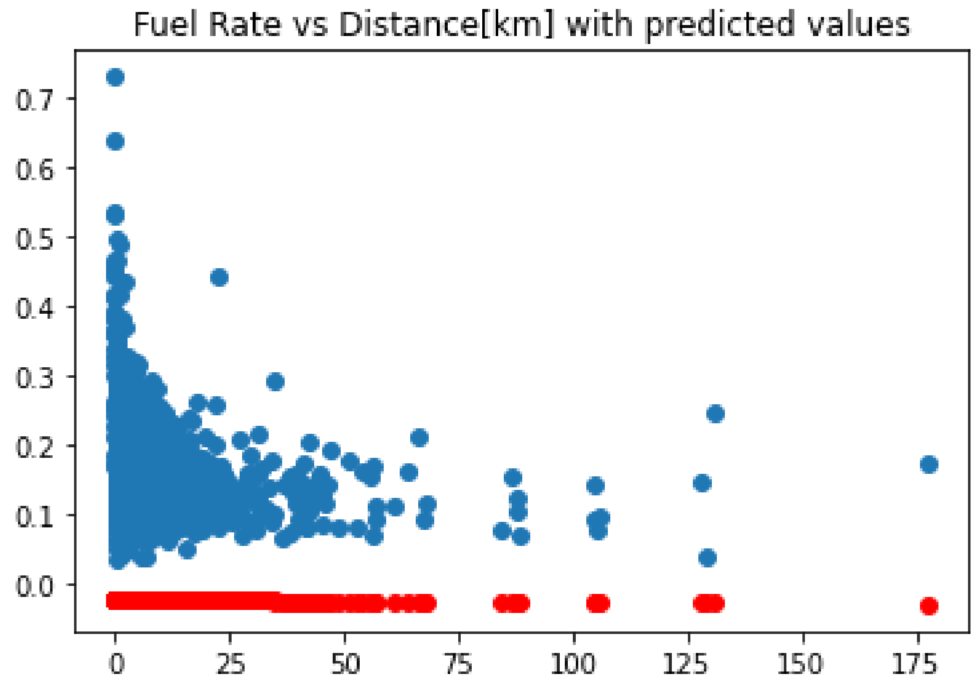
[220]:
```
r_2
```

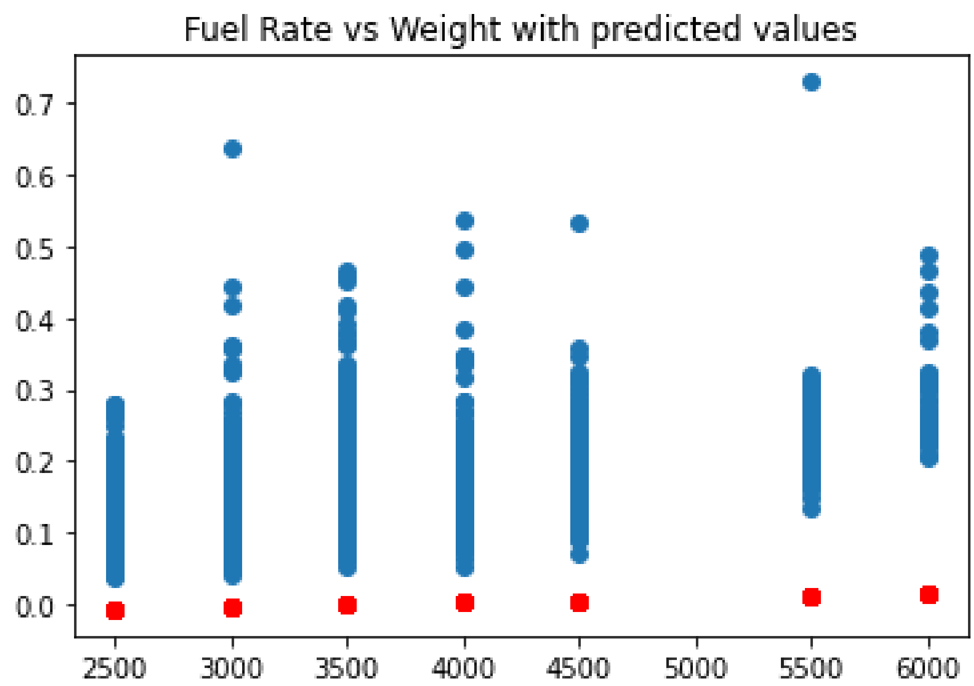
[220]: 0.6123278886732875

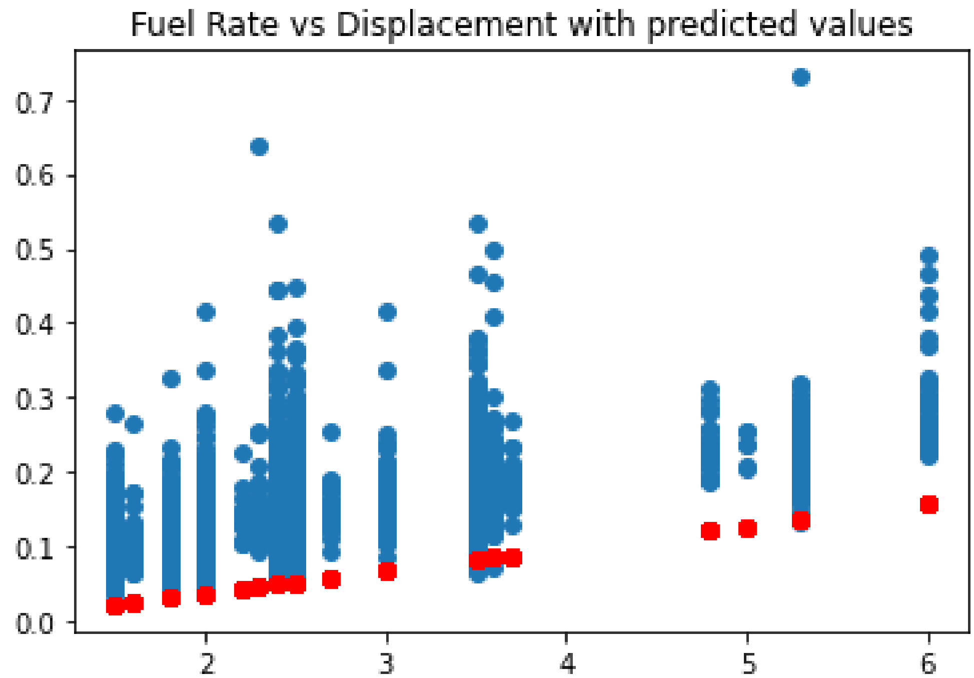
[221]:
```
# now lets draw our plots:
```

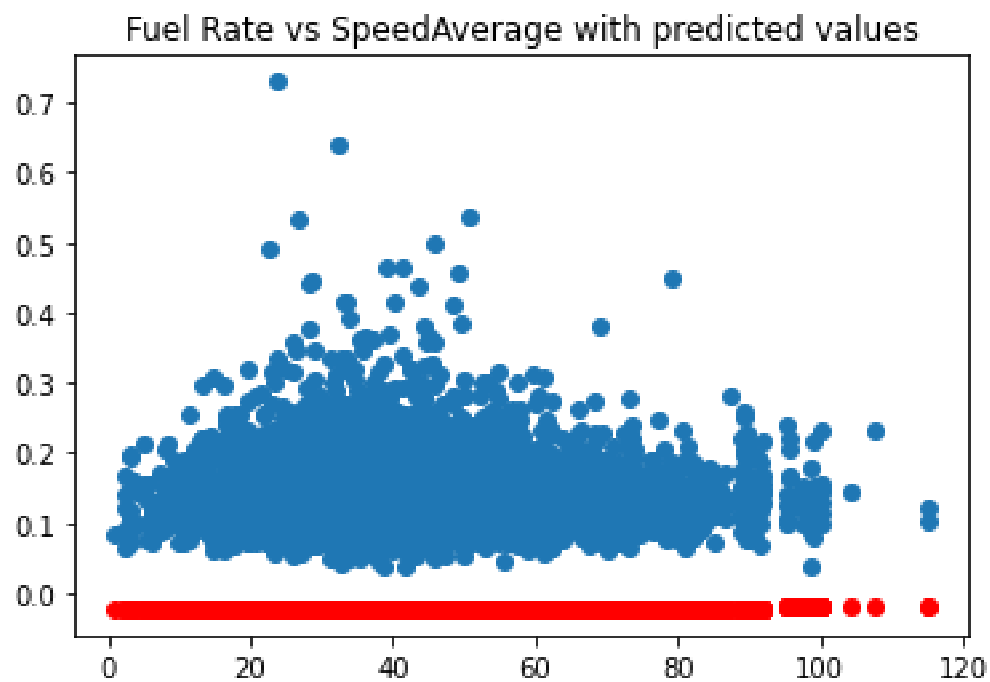
[224]:
```
# single value predictions
for i, feature in enumerate(features):
    plt.scatter(X[:,i], y)
    plt.scatter(X[:,i], model.coef_[0][i]*X[:,i] + model.intercept_[0],␣
 ↪color='red')
    plt.title(f"Fuel Rate vs {feature} with predicted values")
    plt.show()
```

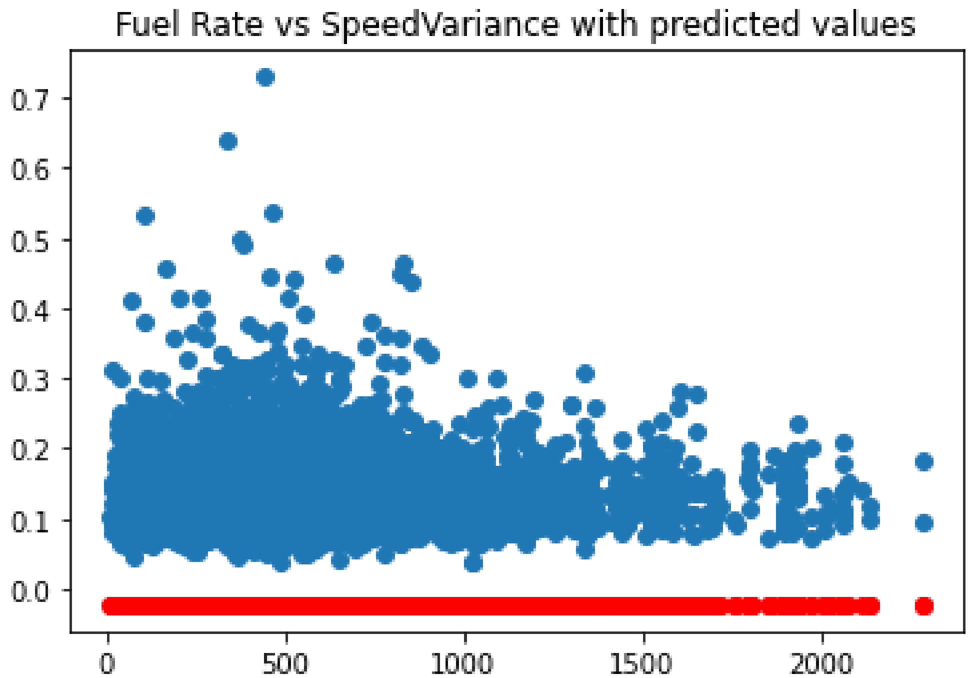



Fuel Rate vs Distance[km] with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
25
50
75
100
125
150
175


Fuel Rate vs Weight with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2500
3000
3500
4000
4500
5000
5500
6000

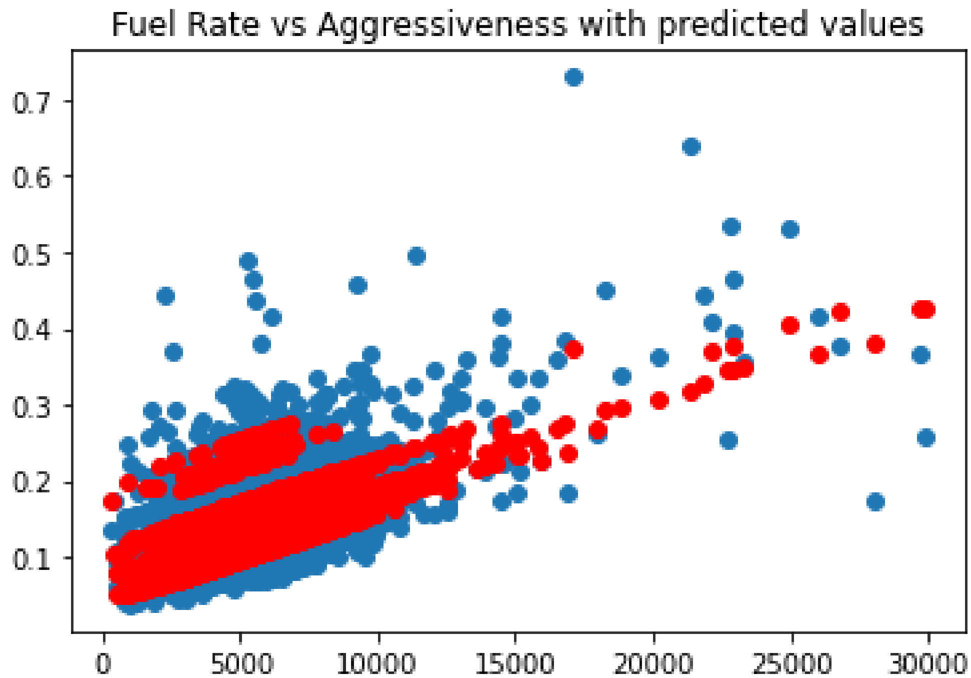

Fuel Rate vs Displacement with predicted values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2
3
4
5
6

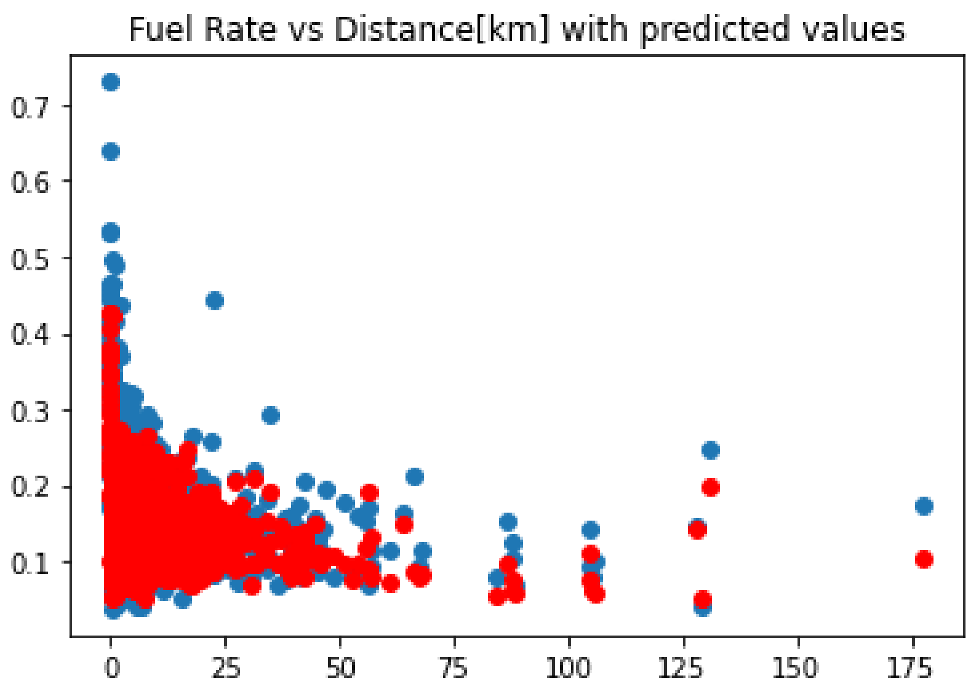

Fuel Rate vs SpeedAverage with predicted values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120

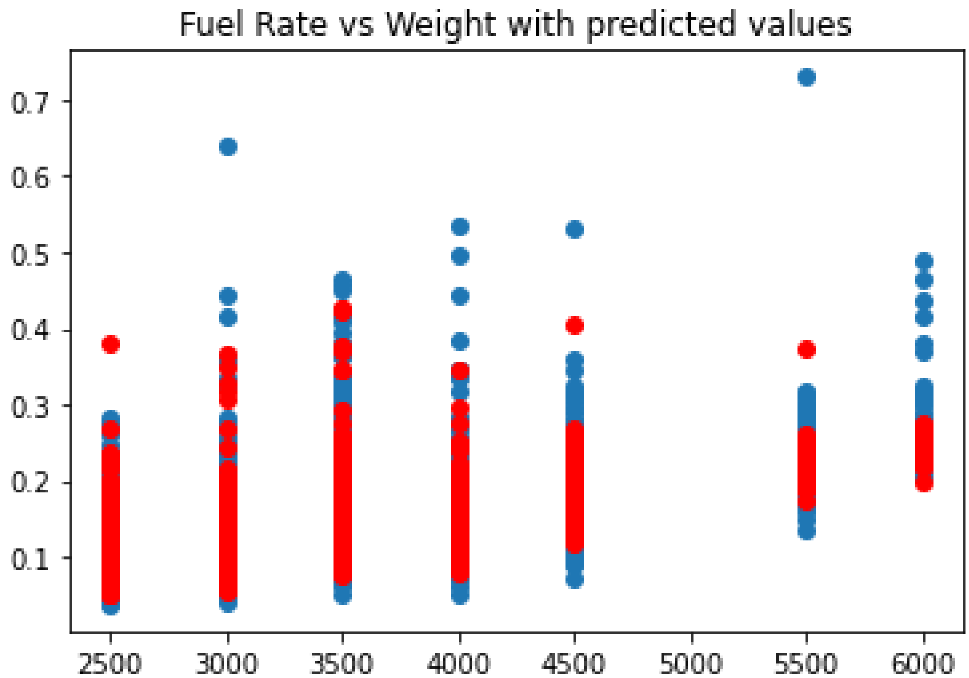


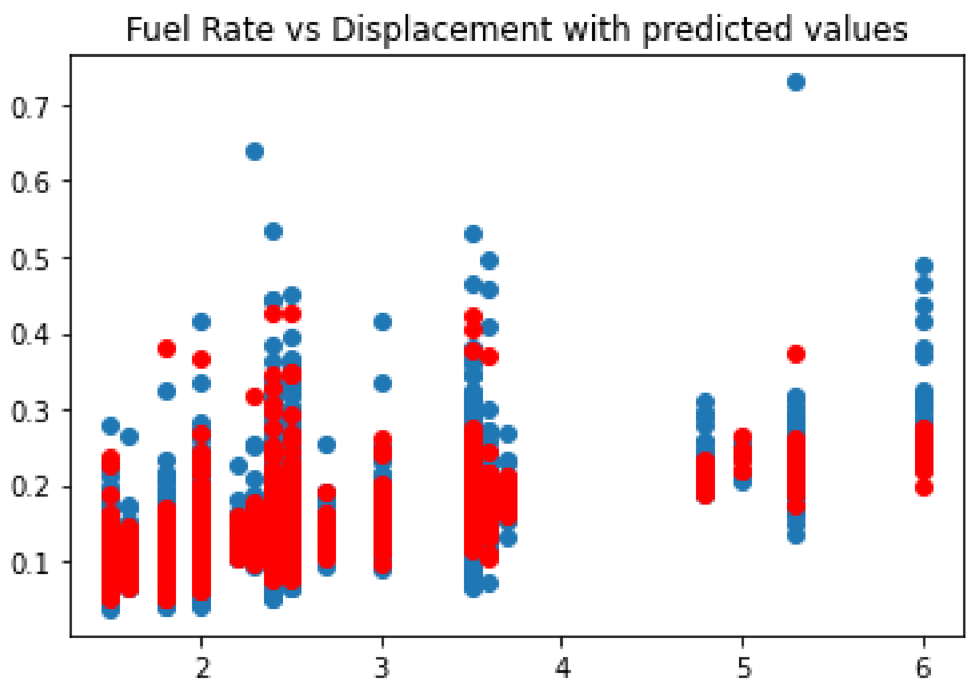
[225]:

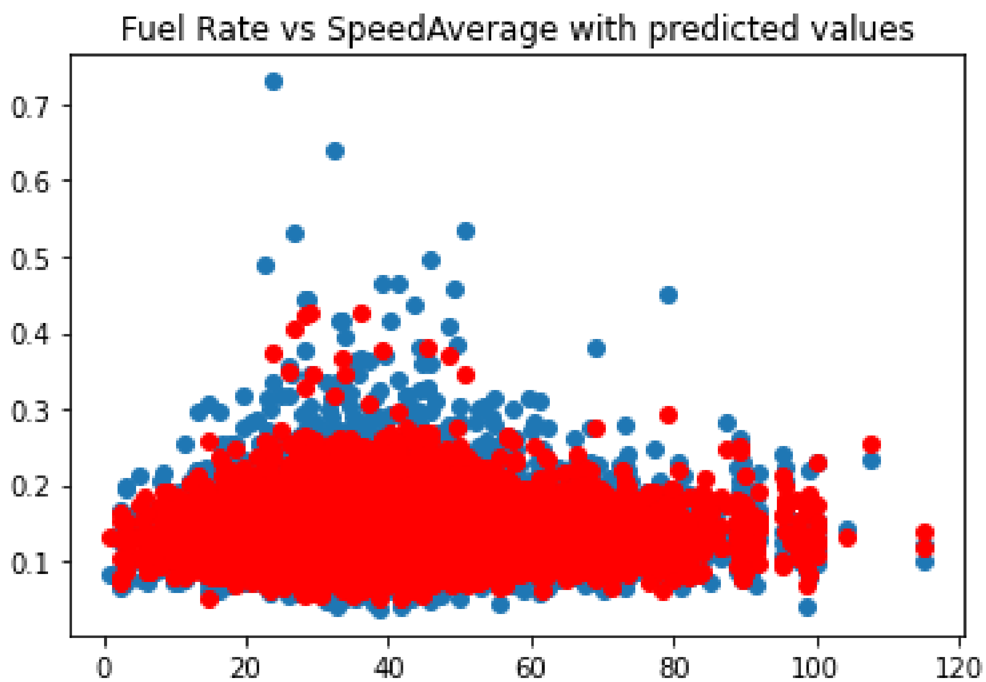
```
#plotting full model values
for i, feature in enumerate(features):
    plt.scatter(X[:,i], y)
    plt.scatter(X[:,i], preds, color='red')
    plt.title(f"Fuel Rate vs {feature} with predicted values")
    plt.show()
```

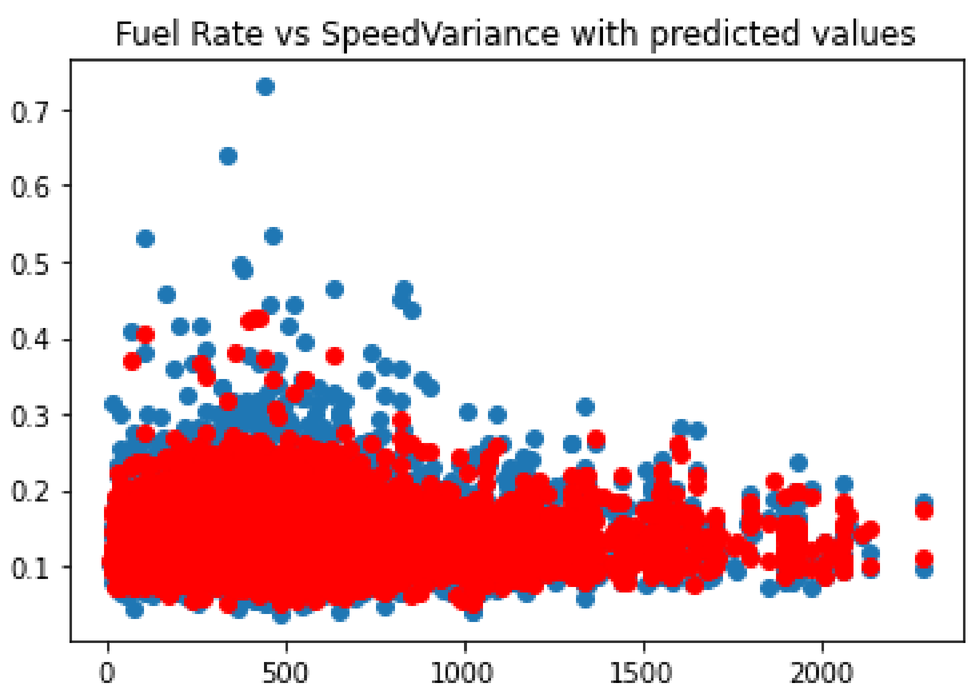

Fuel Rate vs Aggressiveness with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5000
10000
15000
20000
25000
30000


Fuel Rate vs Distance[km] with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
25
50
75
100
125
150
175


Fuel Rate vs Weight with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2500
3000
3500
4000
4500
5000
5500
6000

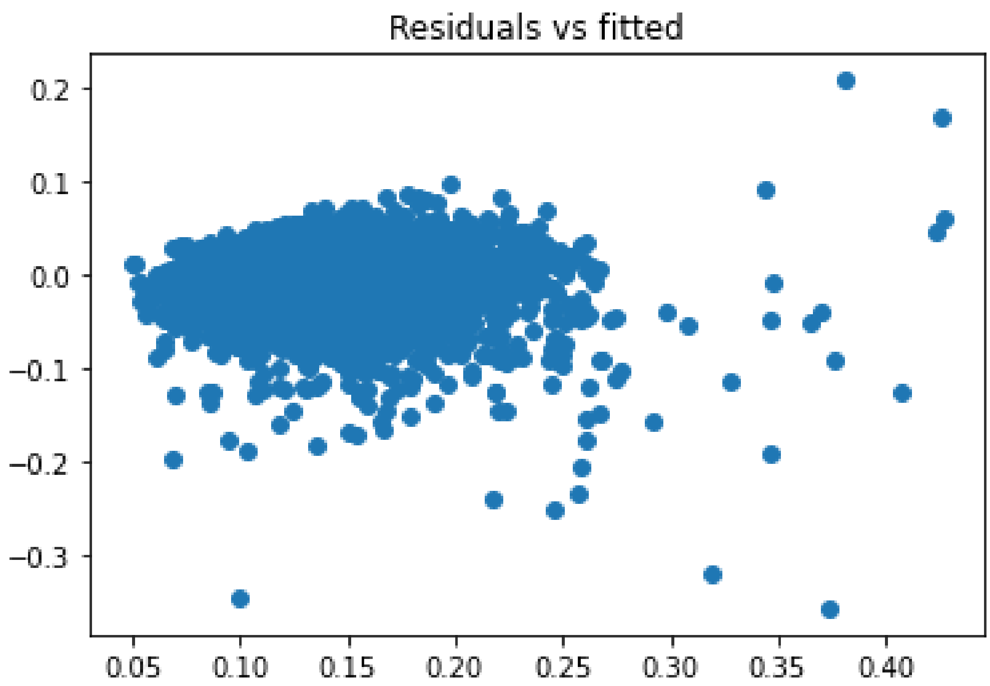

Fuel Rate vs Displacement with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2
3
4
5
6





[226]:
```
# lets plot our residuals:
residuals = (preds - y)
```

```
plt.scatter(preds, residuals)
plt.title('Residuals vs fitted')
plt.show()
```



1.2 Adding Some Interaction Variables

[227]:
```
# thats not a particuarly compelling plot -- lets add some interaction␣
↪variables to our set:
for i, f in enumerate(features):
    print(f"{i} : {f}")
```

```
0 : Aggressiveness
1 : Distance[km]
2 : Weight
3 : Displacement
4 : SpeedAverage
5 : SpeedVariance
```

Lets iteract the following variables:

`Aggressiveness` and `SpeedAverage`

`Aggressiveness` and `Distance[km]`

`Displacement` and `1/Weight` => power to weight

[228]:
```
X.shape
```

[228]:
```
(11529, 6)
```

[229]:
```
X = np.append(X, (X[:,0] * X[:,4]).reshape(-1,1), 1) #aggr and avespeed
X = np.append(X, (X[:,0] * X[:,3]).reshape(-1,1), 1) # aggr and distance
X = np.append(X, (X[:,3] / X[:,2]).reshape(-1,1), 1) # displ and 1/weight
```

[230]:
```
X.shape
```

[230]:
```
(11529, 9)
```

[231]:
```
#lets update our feature list
interaction_features = features + ['Aggr*AveSpeed', 'Aggr*Distance', 'Displ/
↪Weight']
```

[232]:
```
# lets fit our model again:
model = LinearRegression().fit(X,y)
```

[233]:
```
for feature, coef in zip(interaction_features, model.coef_[0]):
    print(f"{feature}: {coef}")
```

```
Aggressiveness: 4.109603391407518e-06
Distance[km]: 1.7205819257853772e-05
Weight: -9.172978419857352e-06
Displacement: 0.03477552361547563
SpeedAverage: -5.035984700663981e-05
SpeedVariance: 1.9055986521698432e-07
Aggr*AveSpeed: 1.2604850684488406e-08
Aggr*Distance: 2.9460774284102315e-06
Displ/Weight: -70.59826830183906
```

[234]:
```
# lets see which value has the highest average impact on our regression:

print('Impact = MEAN * Coef')
for i, vals in enumerate(zip(interaction_features, model.coef_[0])):
    feature, coef = vals
    print(f"{feature}: {X[:,i].mean()*coef}")

# we can actually get the relative percentages here:
print('========= PERCENTAGES =========')

impact = np.array([np.abs(coef * X[:,i].mean()) for i, coef in enumerate(model.
↪coef_[0])])
for feature, i in zip(interaction_features, (impact*100)/impact.sum()):
    print(f"{feature}: {i}")
```

```
Impact = MEAN * Coef
Aggressiveness: 0.021379139574695276
Distance[km]: 9.200587570222754e-05
Weight: -0.030872972082272087
Displacement: 0.08715719293232776
SpeedAverage: -0.001984499793928708
SpeedVariance: 9.780627171411371e-05
Aggr*AveSpeed: 0.00258460756699648
Aggr*Distance: 0.038407058144407265
Displ/Weight: -0.052105052045205595
========= PERCENTAGES =========
Aggressiveness: 9.10989821095411
Distance[km]: 0.03920476591345401
Weight: 13.15532985583848
Displacement: 37.138686203526134
SpeedAverage: 0.8456182747292637
SpeedVariance: 0.04167638162403438
Aggr*AveSpeed: 1.1013311255270803
Aggr*Distance: 16.365690913579872
Displ/Weight: 22.202564268307555
```

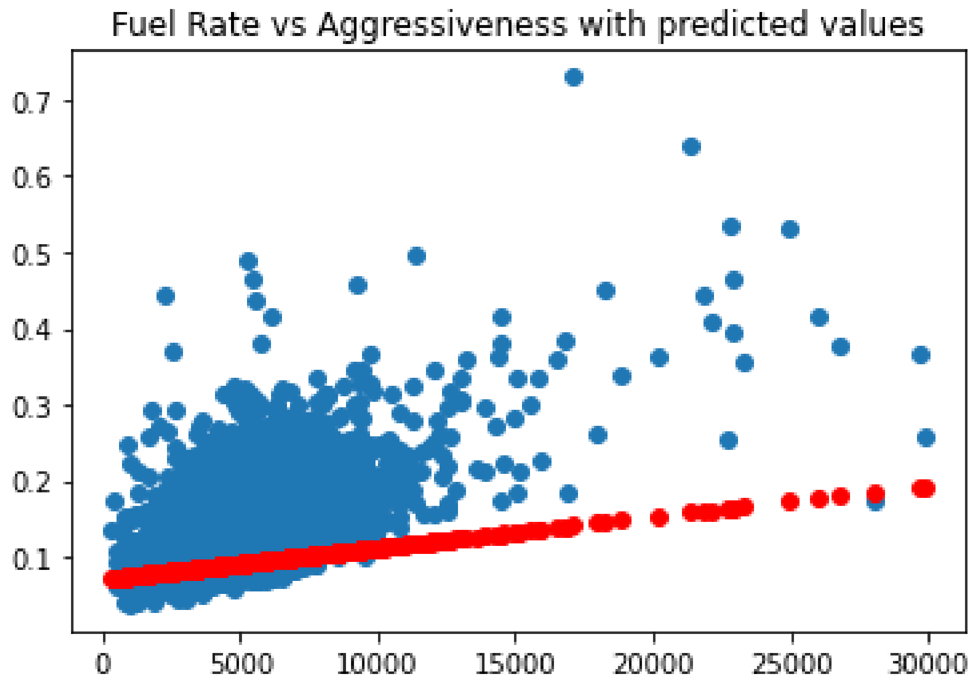
[235]:

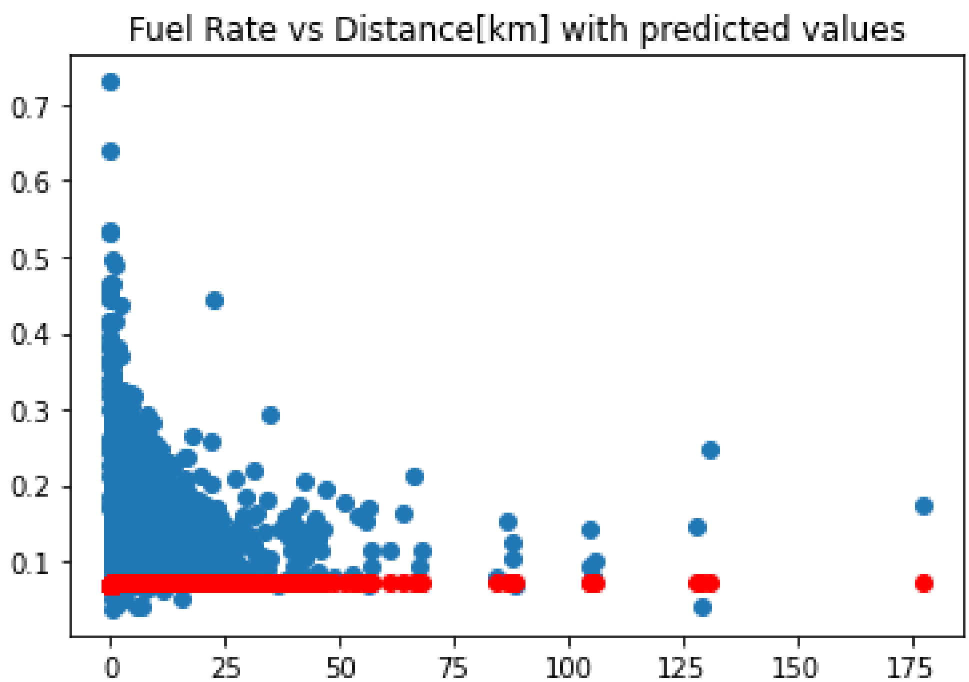
```
# now lets compute r^2:
# WOW WE have an almost 0 impact on r^2 -- surprising
preds = model.predict(X)
r_2 = 1 - (((y - preds)**2).sum())/((y - y.mean())**2).sum()
print(r_2)
```

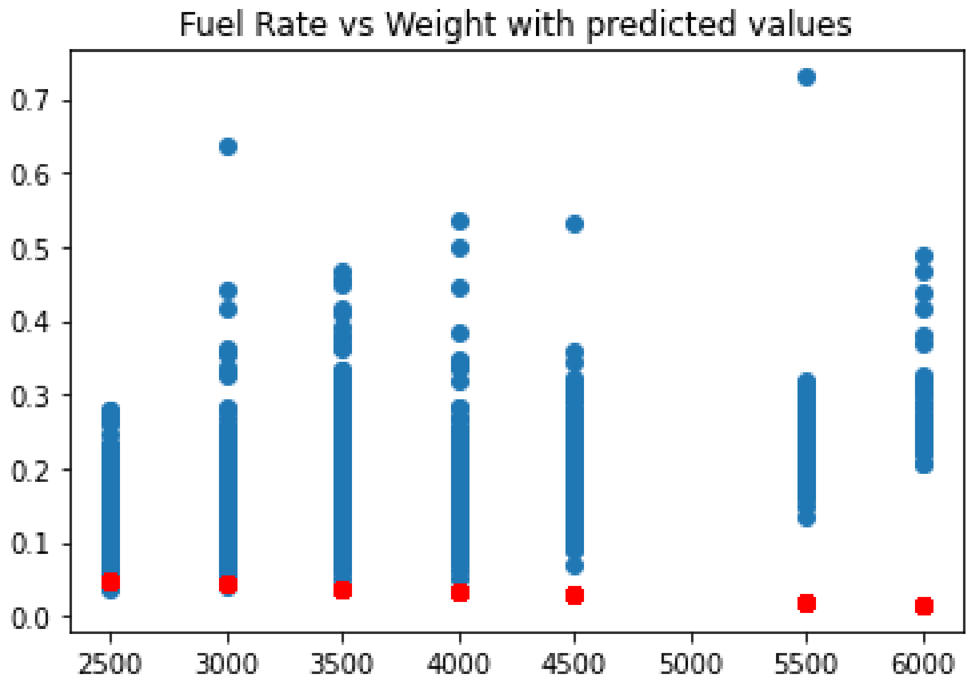
```
0.6200223135986707
```

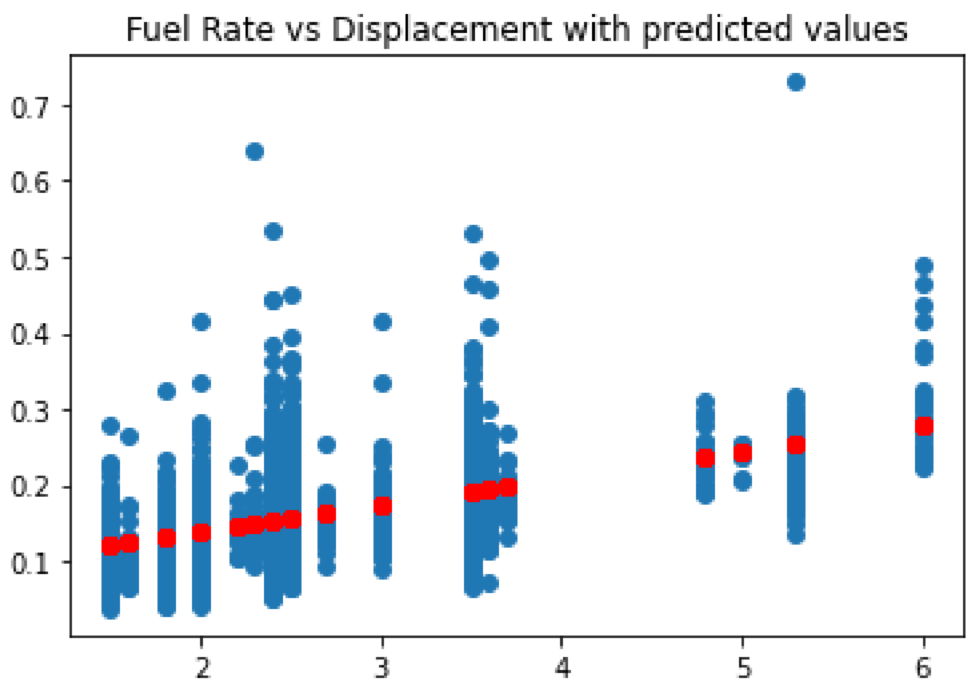
[236]:

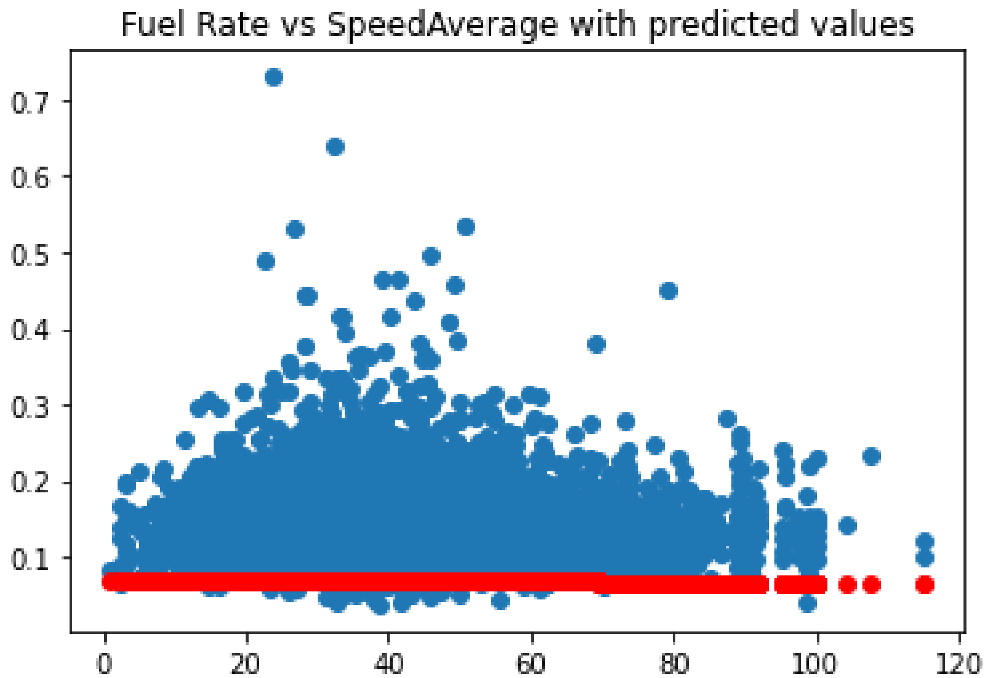
```
# single value predictions
for i, feature in enumerate(interaction_features):
    plt.scatter(X[:,i], y)
    plt.scatter(X[:,i], model.coef_[0][i]*X[:,i] + model.intercept_[0],␣
 ↪color='red')
    plt.title(f"Fuel Rate vs {feature} with predicted values")
    plt.show()
```

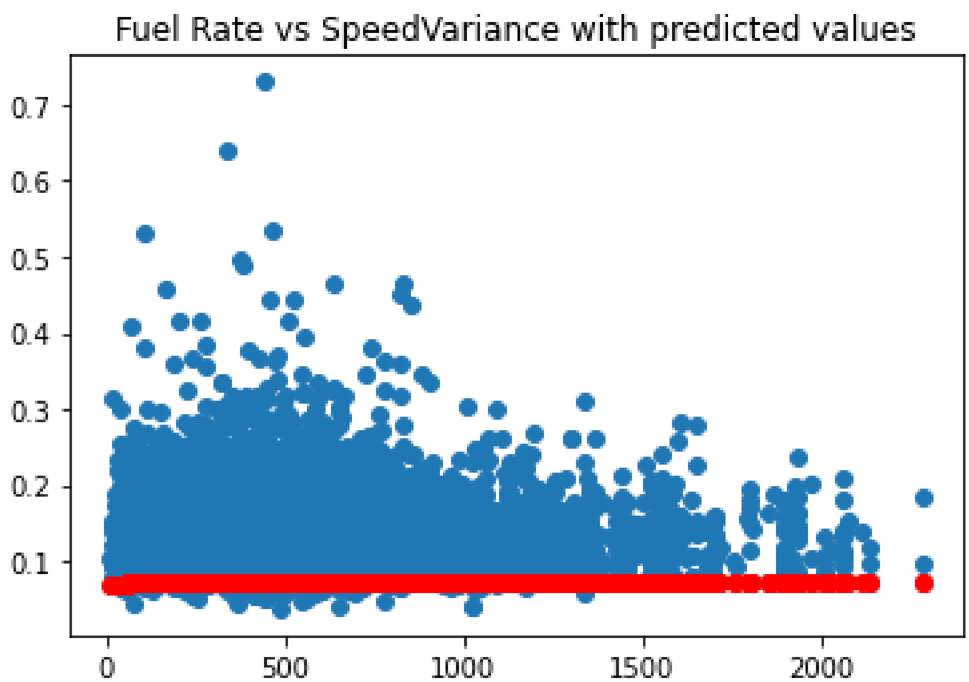

Fuel Rate vs Aggressiveness with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5000
10000
15000
20000
25000
30000

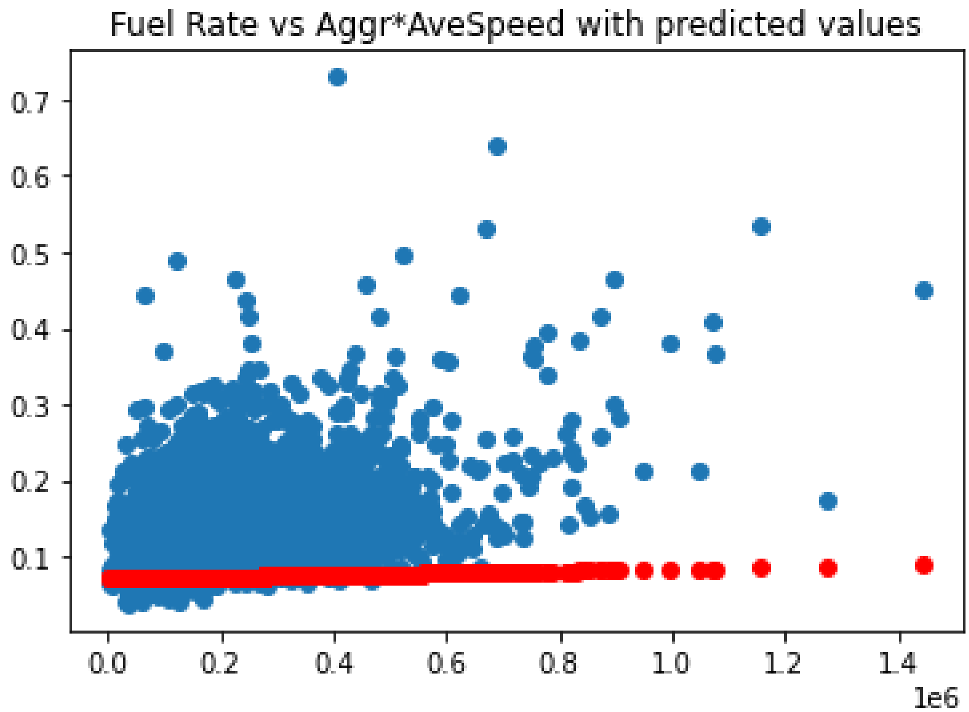

Fuel Rate vs Distance[km] with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
25
50
75
100
125
150
175

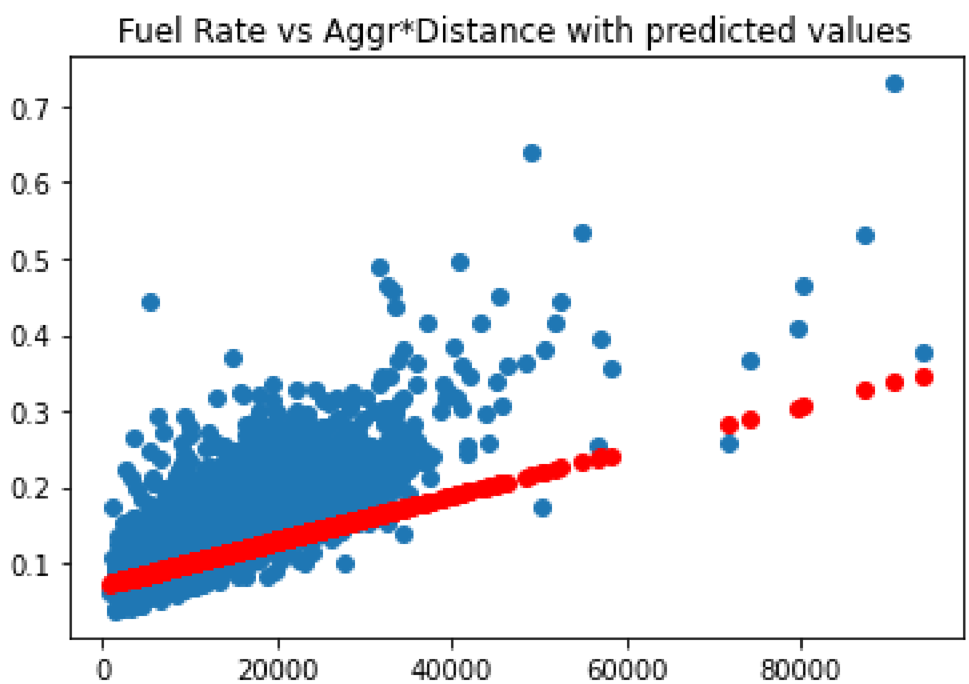

Fuel Rate vs Weight with predicted values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2500
3000
3500
4000
4500
5000
5500
6000

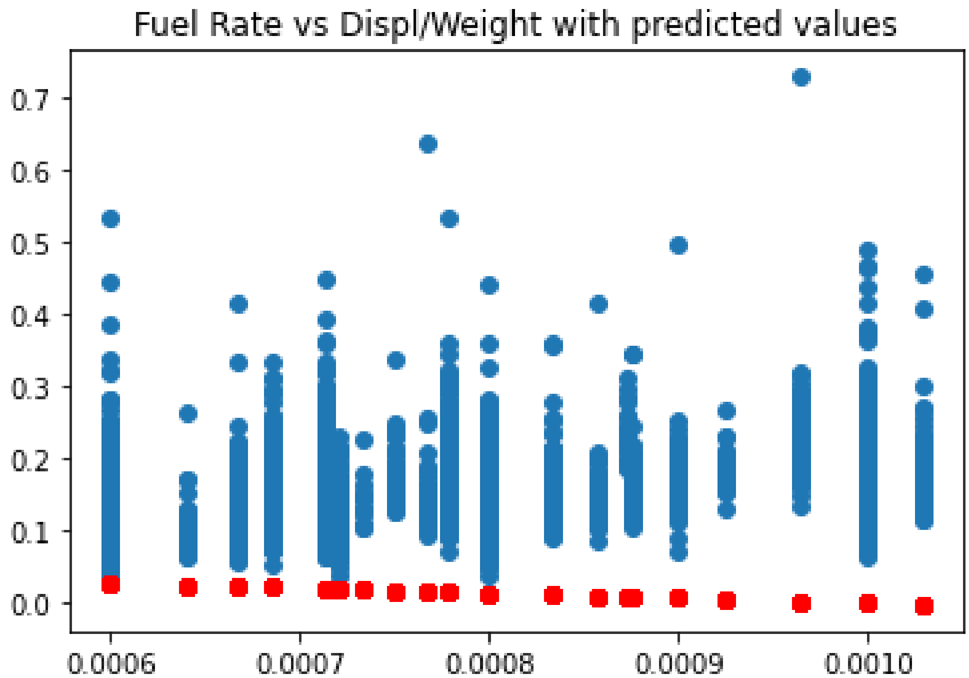

Fuel Rate vs Displacement with predicted values
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2
3
4
5
6


Fuel Rate vs SpeedAverage with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120


Fuel Rate vs SpeedVariance with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000

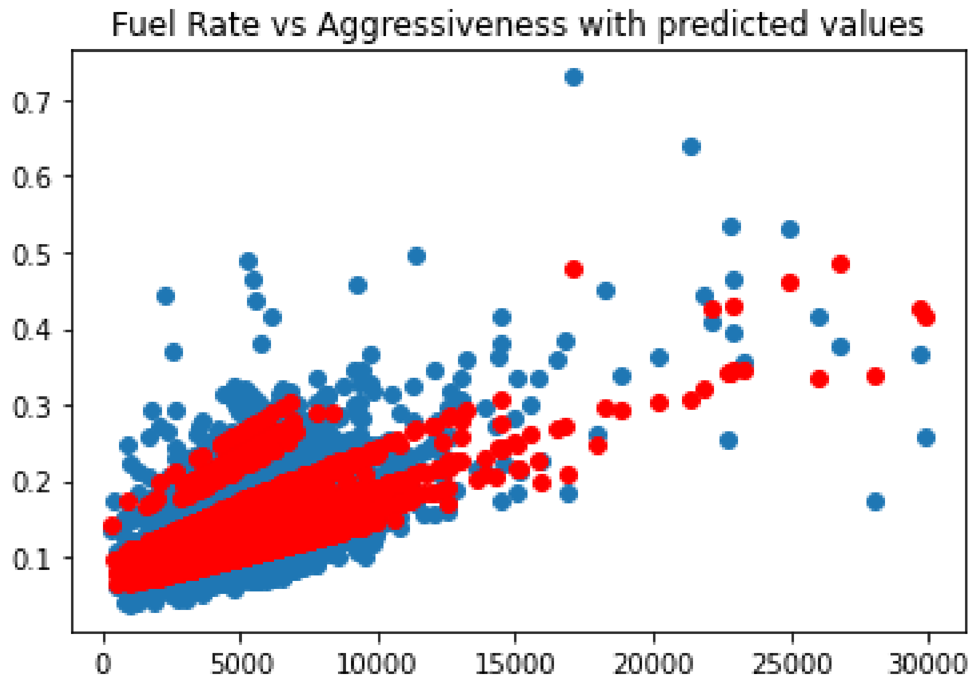

Fuel Rate vs Aggr*AveSpeed with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6

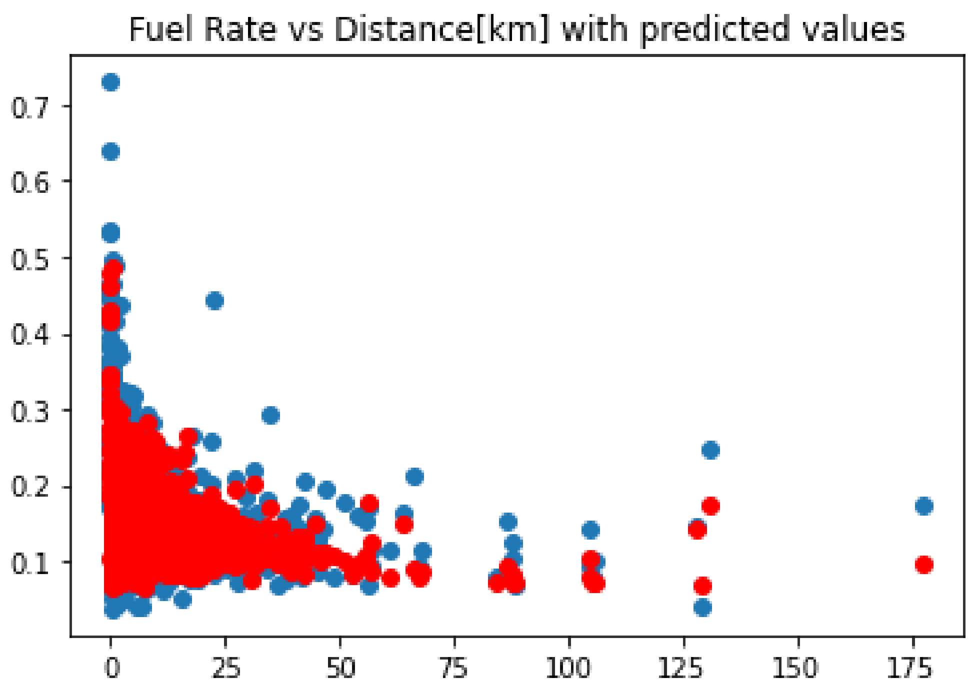

Fuel Rate vs Aggr*Distance with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20000
40000
60000
80000

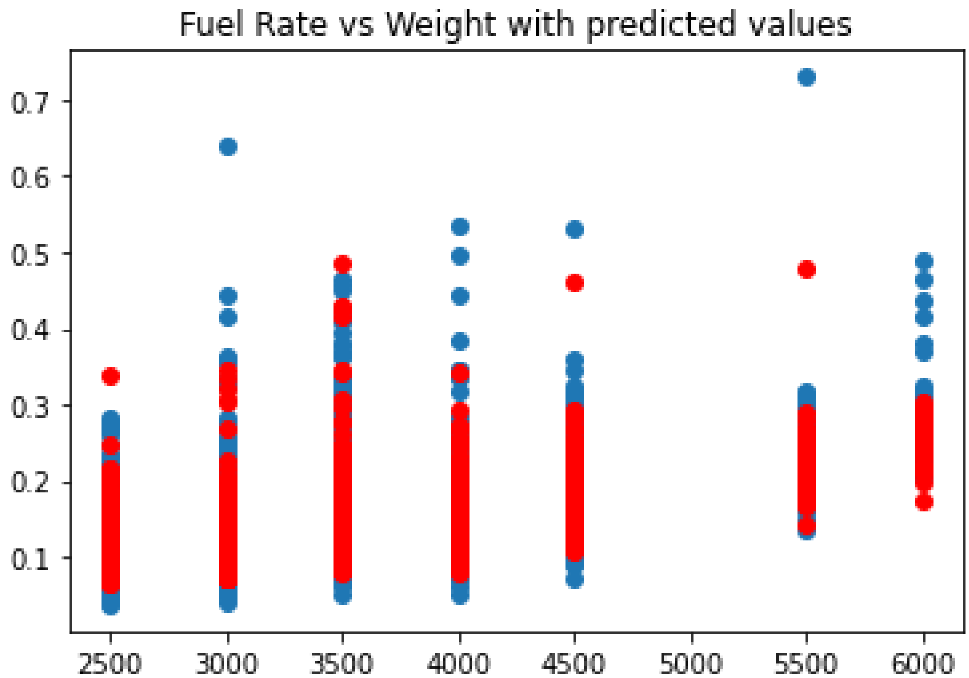


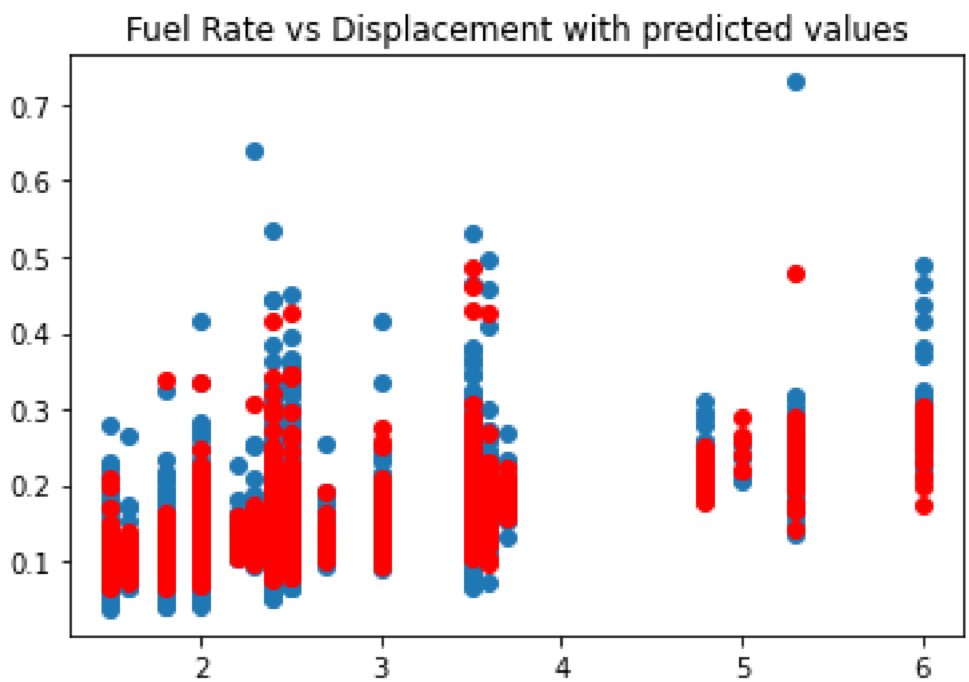
[238]:

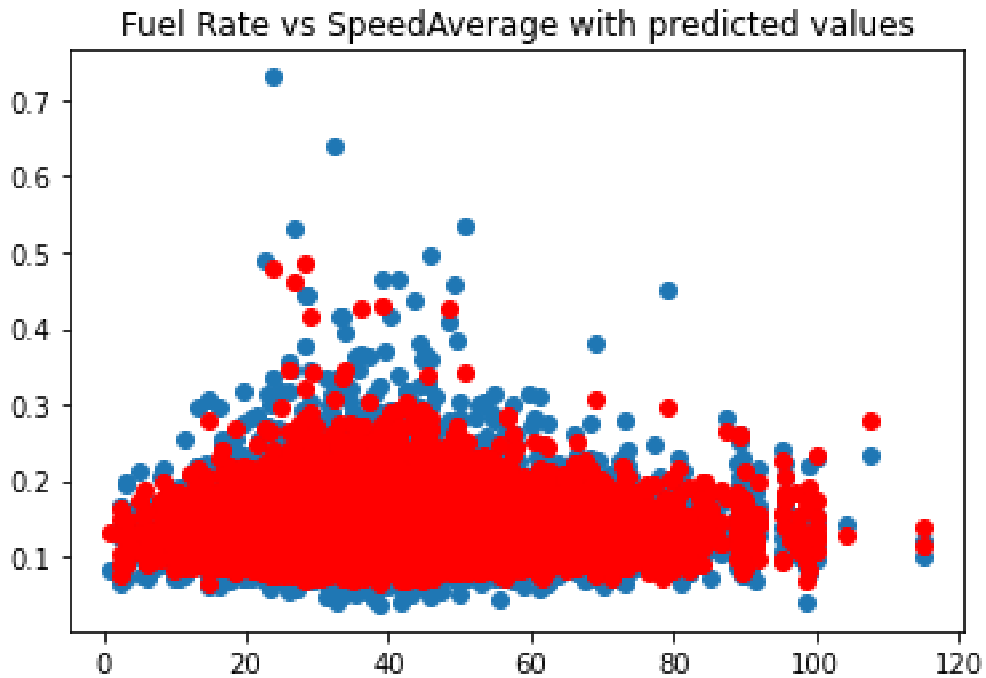
```
#plotting full model values
for i, feature in enumerate(interaction_features):
    plt.scatter(X[:,i], y)
    plt.scatter(X[:,i], preds, color='red')
    plt.title(f"Fuel Rate vs {feature} with predicted values")
    plt.show()
```

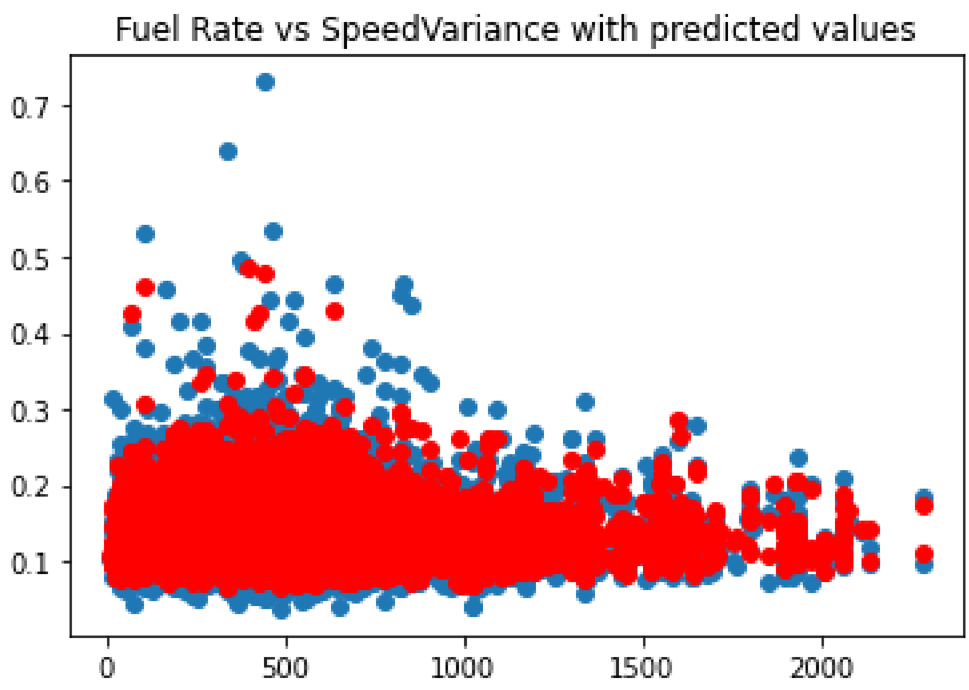

Fuel Rate vs Aggressiveness with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5000
10000
15000
20000
25000
30000

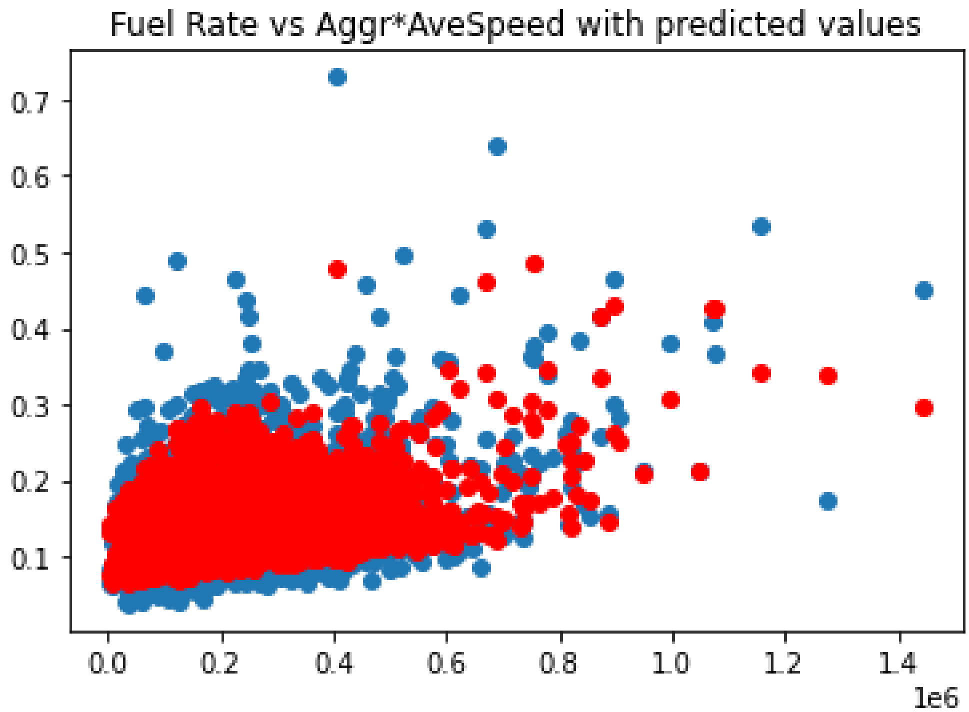

Fuel Rate vs Distance[km] with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
25
50
75
100
125
150
175

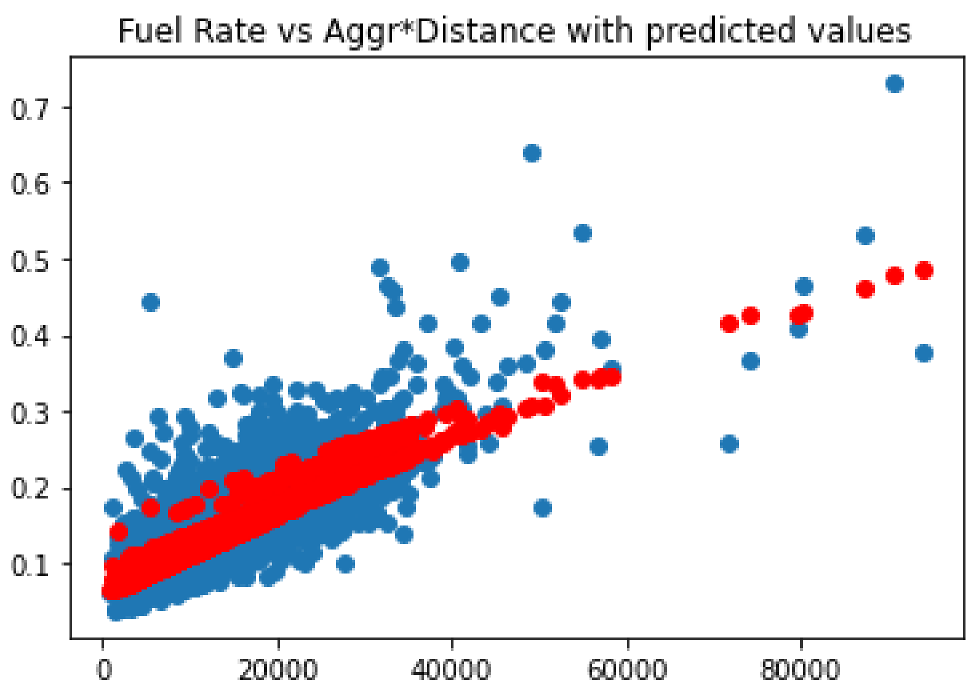

Fuel Rate vs Weight with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2500
3000
3500
4000
4500
5000
5500
6000

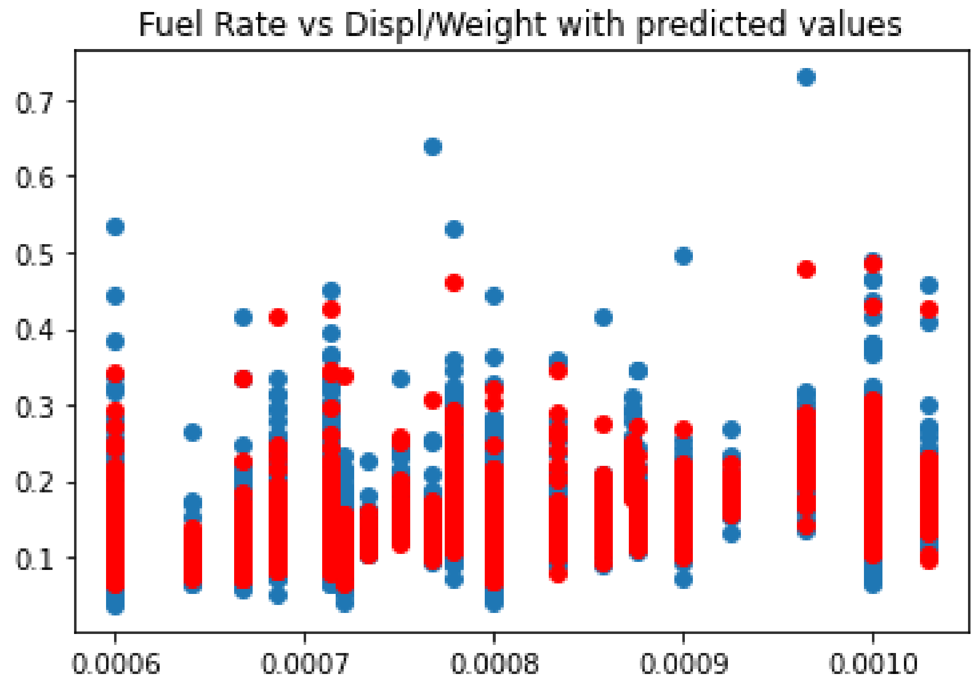

Fuel Rate vs Displacement with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2
3
4
5
6


Fuel Rate vs SpeedAverage with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120

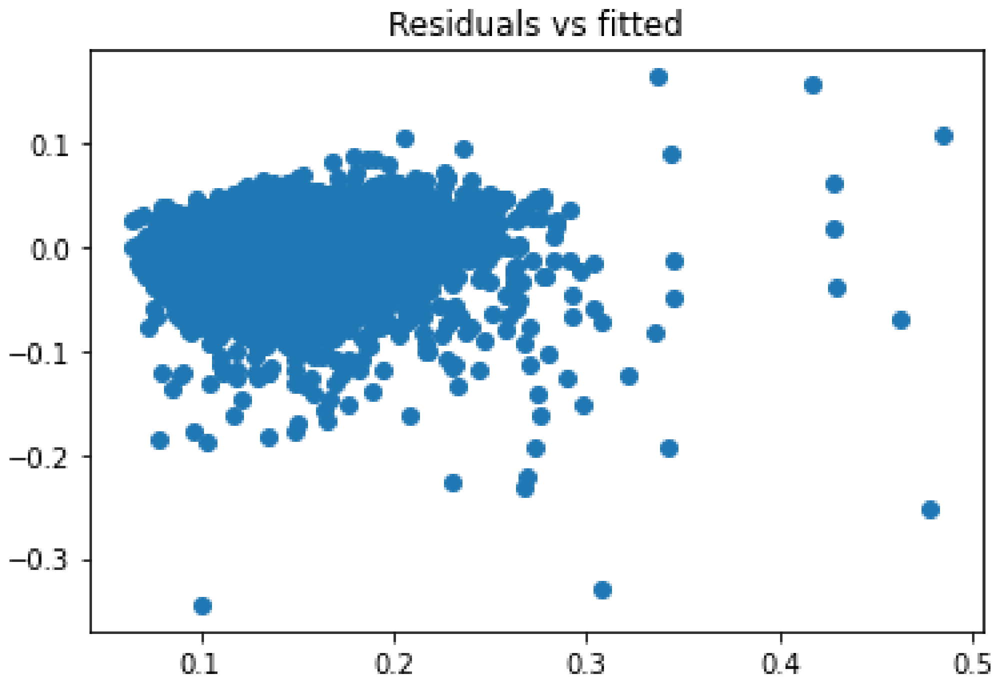

Fuel Rate vs SpeedVariance with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000


Fuel Rate vs Aggr*AveSpeed with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6


Fuel Rate vs Aggr*Distance with predicted values
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20000
40000
60000
80000



[239]:
```
# resuidual vs fitted plots!
residuals = (preds - y)
plt.scatter(preds, residuals)
plt.title('Residuals vs fitted')
plt.show() # we have the same issue with this plot where our residuals vs␣
↪fitted are still clumpy
```



1.3 Lets run a larger interaction variable test:

[255]:
```
# reset the data
X = np.array(data.loc[:, features])
y = np.array(data.loc[:,target]).reshape(-1,1)
```

[256]:
```
X.shape
```

[256]: (11529, 6)

[257]:
```
y.shape
```

[257]: (11529, 1)

[258]:
```
# lets make all the interaction variables:
all_interaction = features.copy()
for i in range(len(features)):
    for j in range(i+1, len(features)):
        X = np.append(X, (X[:, i] * X[:, j]).reshape(-1,1), 1)
        all_interaction.append(f"{features[i]}*{features[j]}")
```

[259]:
```
all_interaction
```

[259]:

```
['Aggressiveness',
 'Distance[km]',
 'Weight',
 'Displacement',
 'SpeedAverage',
 'SpeedVariance',
 'Aggressiveness*Distance[km]',
 'Aggressiveness*Weight',
 'Aggressiveness*Displacement',
 'Aggressiveness*SpeedAverage',
 'Aggressiveness*SpeedVariance',
 'Distance[km]*Weight',
 'Distance[km]*Displacement',
 'Distance[km]*SpeedAverage',
 'Distance[km]*SpeedVariance',
 'Weight*Displacement',
 'Weight*SpeedAverage',
 'Weight*SpeedVariance',
 'Displacement*SpeedAverage',
 'Displacement*SpeedVariance',
 'SpeedAverage*SpeedVariance']
```

[260]:

```
X.shape
```

[260]:

```
(11529, 21)
```

[261]:

```
model = LinearRegression().fit(X, y)
```

[264]:

```
for feature, coef in zip(all_interaction, model.coef_[0]):
    print(f"{feature}: {coef}")
```

```
Aggressiveness: 2.5022926173706823e-06
Distance[km]: 0.00030098050612477783
Weight: -1.1893202490620523e-05
Displacement: 0.010147303150106018
SpeedAverage: 6.144613174886634e-05
SpeedVariance: 1.974361128572301e-07
Aggressiveness*Distance[km]: -3.8251066330590243e-07
Aggressiveness*Weight: 1.7675447132677472e-09
Aggressiveness*Displacement: 1.6223401111833852e-06
Aggressiveness*SpeedAverage: 1.7993876257149626e-08
Aggressiveness*SpeedVariance: -6.893053204528441e-10
Distance[km]*Weight: 3.375073559130909e-07
Distance[km]*Displacement: -0.00022565392091916604
Distance[km]*SpeedAverage: 5.175361508629751e-06
Distance[km]*SpeedVariance: -1.7215678674891726e-09
Weight*Displacement: 3.2538600924069544e-06
```

```
Weight*SpeedAverage: -1.4078378054548045e-08
Weight*SpeedVariance: 4.779569972704856e-10
Displacement*SpeedAverage: -3.919495876954216e-05
Displacement*SpeedVariance: 1.8208014369773895e-06
SpeedAverage*SpeedVariance: -4.7788005972552274e-08
```

[265]:

```python
# lets see which value has the highest average impact on our regression:

print('Impact = MEAN * Coef')
for i, vals in enumerate(zip(all_interaction, model.coef_[0])):
    feature, coef = vals
    print(f"{feature}: {X[:,i].mean()*coef}")

# we can actually get the relative percentages here:
print('========= PERCENTAGES =========')

impact = np.array([np.abs(coef * X[:,i].mean()) for i, coef in enumerate(model.
 ↪coef_[0])])
for feature, i in zip(all_interaction, (impact*100)/impact.sum()):
    print(f"{feature}: {i}")
```

```
Impact = MEAN * Coef
Aggressiveness: 0.013017524570704037
Distance[km]: 0.0016094540236826912
Weight: -0.04002827562167602
Displacement: 0.025431981073119 82
SpeedAverage: 0.002421370259072949
SpeedVariance: 0.00010133555708755586
Aggressiveness*Distance[km]: -0.009542428587891507
Aggressiveness*Weight: 0.031002679734935045
Aggressiveness*Displacement: 0.021149923073762494
Aggressiveness*SpeedAverage: 0.003689619964404585
Aggressiveness*SpeedVariance: -0.0018368910594713528
Distance[km]*Weight: 0.006168451356207922
Distance[km]*Displacement: -0.003084888067653157
Distance[km]*SpeedAverage: 0.0010893730424327756
Distance[km]*SpeedVariance: -4.766858874831378e-06
Weight*Displacement: 0.02881071700321241
Weight*SpeedAverage: -0.001866221008820074
Weight*SpeedVariance: 0.0008241984075872981
Displacement*SpeedAverage: -0.0038689221787572207
Displacement*SpeedVariance: 0.002339180110091627 4
SpeedAverage*SpeedVariance: -0.0010369376461777547
========= PERCENTAGES =========
Aggressiveness: 6.543931361660887
Distance[km]: 0.8090752280530229
Weight: 20.122281065893784
Displacement: 12.784699397310199
```

```
SpeedAverage: 1.2172268767750087
SpeedVariance: 0.05094155393944858
Aggressiveness*Distance[km]: 4.796994802164125
Aggressiveness*Weight: 15.585098926530561
Aggressiveness*Displacement: 10.632101683186658
Aggressiveness*SpeedAverage: 1.8547781236391274
Aggressiveness*SpeedVariance: 0.9234082061255259
Distance[km]*Weight: 3.100890807889174
Distance[km]*Displacement: 1.550778388278212
Distance[km]*SpeedAverage: 0.5476296494163669
Distance[km]*SpeedVariance: 0.0023963079246127894
Weight*Displacement: 14.483195597229951
Weight*SpeedAverage: 0.9381524206907869
Weight*SpeedVariance: 0.414325917216198
Displacement*SpeedAverage: 1.9449136465140398
Displacement*SpeedVariance: 1.1759097514938426
SpeedAverage*SpeedVariance: 0.5212702880684681
```

[266]:
```
# now lets compute r^2:
# WOW WE have an almost 0 impact on r^2 -- surprising
preds = model.predict(X)
r_2 = 1 - (((y - preds)**2).sum())/((y - y.mean())**2).sum()
print(r_2)
```

```
0.634915533525551
```

1.4 Optimizing our Linear Regression with Smart Variable Choices

[ ]:
```
```