In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
import statsmodels.tsa.api as smt
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import PolynomialFeatures
```

In [2]:

```
data= pd.read_excel('3D Objects/data set 04012023.xlsx')
data.head()
```

Out[2]:

	Vehicle	AT	BC	centre	MP	EC	DT	Radius(m)	Dc	Lane Width(m)	...	AT-BC	BC-MP	MP-EC	
0	Class 1	92.700	58.16	62.680000	55.46	74.42	85.5	100	57.272727	3.50	...	-0.472902	-2.166683	-1.969426	-0.3
1	Class 1	102.400	101.50	96.266667	93.40	93.90	98.4	119	48.128342	3.50	...	-0.014160	-1.218125	0.072261	0.(
2	Class 1	85.585	84.00	68.566667	60.50	61.20	69.8	172	33.298097	3.25	...	-0.020740	-1.679601	0.042137	0.(
3	Class 1	100.400	100.00	83.433333	74.00	76.30	90.3	220	26.033058	3.50	...	-0.005154	-1.342593	0.102591	0.1
4	Class 1	95.610	94.71	88.576667	84.91	86.11	97.1	297	19.283747	3.25	...	-0.018881	-0.565932	0.065980	0.2

5 rows × 23 columns

In [3]:

```
#data cleaning
data=data.drop(['AT-BC','BC-MP','MP-EC','EC-DT'],axis=1)
data.head()
data.sort_values(by='Vehicle')
data.isnull()
data.fillna(0,inplace = True)
#adding new column and calculations
v85=((data['AT']+data['BC']+data['MP']+data['EC']+data['DT'])) * (0.85)
v85=v85/5
data['v85']= v85
v85_C=((data['BC']+data['MP']+data['EC']) * 0.85) / 3
data['v85_C']=v85_C
v85_T=((data['AT']+data['DT']) * 0.85) /2
data['v85_T']=v85_T
data.head(5)
```

Out[3]:

	Lane	Speed	acc	tangent

	Vehicle	AT	BC	centre	MP	EC	DT	Radius(m)	Dc	Width(m)	...	Limit(kph)	decc rate	rate	length
0	Class 1	92.700	58.16	62.680000	55.46	74.42	85.5	100	57.272727	3.50	...	100	-1.228385	0.0	300
1	Class 1	102.400	101.50	96.266667	93.40	93.90	98.4	119	48.128342	3.50	...	100	-0.273313	0.0	700
2	Class 1	85.585	84.00	68.566667	60.50	61.20	69.8	172	33.298097	3.25	...	70	-0.392819	0.0	500
3	Class 1	100.400	100.00	83.433333	74.00	76.30	90.3	220	26.033058	3.50	...	100	-0.273795	0.0	750
4	Class 1	95.610	94.71	88.576667	84.91	86.11	97.1	297	19.283747	3.25	...	70	-0.074222	0.0	600

5 rows × 22 columns

In [ ]:

In [4]:

```
data.info()
d85= data['decc rate'] * 0.85
data['d85']= d85
a85= data['acc rate'] * 0.85
data['a85']=a85
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Vehicle          65 non-null     object
 1   AT               65 non-null     float64
 2   BC               65 non-null     float64
 3   centre           65 non-null     float64
 4   MP               65 non-null     float64
 5   EC               65 non-null     float64
 6   DT               65 non-null     float64
 7   Radius(m)        65 non-null     int64
 8   Dc               65 non-null     float64
 9   Lane Width(m)    65 non-null     float64
 10  Curve Length(m)  65 non-null     float64
 11  Deflection Angle  65 non-null    float64
 12  Speed Limit(kph)  65 non-null    int64
 13  decc rate        65 non-null     float64
 14  acc rate         65 non-null     float64
 15  tangent length   65 non-null     int64
 16  stl              65 non-null     int64
 17  ∆v85             65 non-null     float64
 18  density          65 non-null     float64
 19  v85              65 non-null     float64
 20  v85_C            65 non-null     float64
 21  v85_T            65 non-null     float64
dtypes: float64(17), int64(4), object(1)
memory usage: 11.3+ KB
```

In [5]:

```
vehicle_class1= data.loc[data.Vehicle =='Class 1']
vehicle_class2= data.loc[data.Vehicle =='Class 2']
vehicle_class3= data.loc[data.Vehicle =='Class 3']
vehicle_class4= data.loc[data.Vehicle =='Class 4']
vehicle_class5= data.loc[data.Vehicle =='Class 5']
```

vehicle_class1

Out[5]:

	Vehicle	AT	BC	centre	MP	EC	DT	Radius(m)	Dc	Lane Width(m)	...	acc rate	tangent length	stl	Δv85
0	Class 1	92.700	58.16	62.680000	55.46	74.42	85.5	100	57.272727	3.50	...	0.000000	300	300	25.517000
1	Class 1	102.400	101.50	96.266667	93.40	93.90	98.4	119	48.128342	3.50	...	0.000000	700	700	13.033333
2	Class 1	85.585	84.00	68.566667	60.50	61.20	69.8	172	33.298097	3.25	...	0.000000	500	500	0.311667
3	Class 1	100.400	100.00	83.433333	74.00	76.30	90.3	220	26.033058	3.50	...	0.000000	750	750	12.129500
4	Class 1	95.610	94.71	88.576667	84.91	86.11	97.1	297	19.283747	3.25	...	0.000000	600	600	-9.231000
5	Class 1	110.000	109.70	94.666667	86.70	87.60	103.6	468	12.237762	3.50	...	0.000000	500	500	5.213333
6	Class 1	110.900	110.30	88.700000	77.30	78.50	108.3	485	11.808810	3.25	...	0.000000	750	750	5.978333
7	Class 1	94.180	93.88	96.626667	97.90	98.10	99.0	597	9.593422	3.50	...	0.028877	700	700	3.428333
8	Class 1	97.100	95.90	94.700000	93.70	94.50	95.5	830	6.900329	3.50	...	0.000000	500	500	4.992333
9	Class 1	103.700	103.50	99.666667	97.70	97.80	102.2	980	5.844156	3.50	...	0.000000	500	500	4.301000
10	Class 1	83.600	83.40	83.233333	83.10	83.20	83.5	1300	4.405594	3.50	...	0.000000	600	600	0.226667
11	Class 1	87.000	86.80	86.633333	86.50	86.60	86.9	1110	5.159705	3.50	...	0.000000	350	350	14.988333
12	Class 1	101.200	100.30	95.066667	92.20	92.70	97.2	116	49.373041	3.25	...	0.000000	800	800	1.989000

13 rows × 24 columns

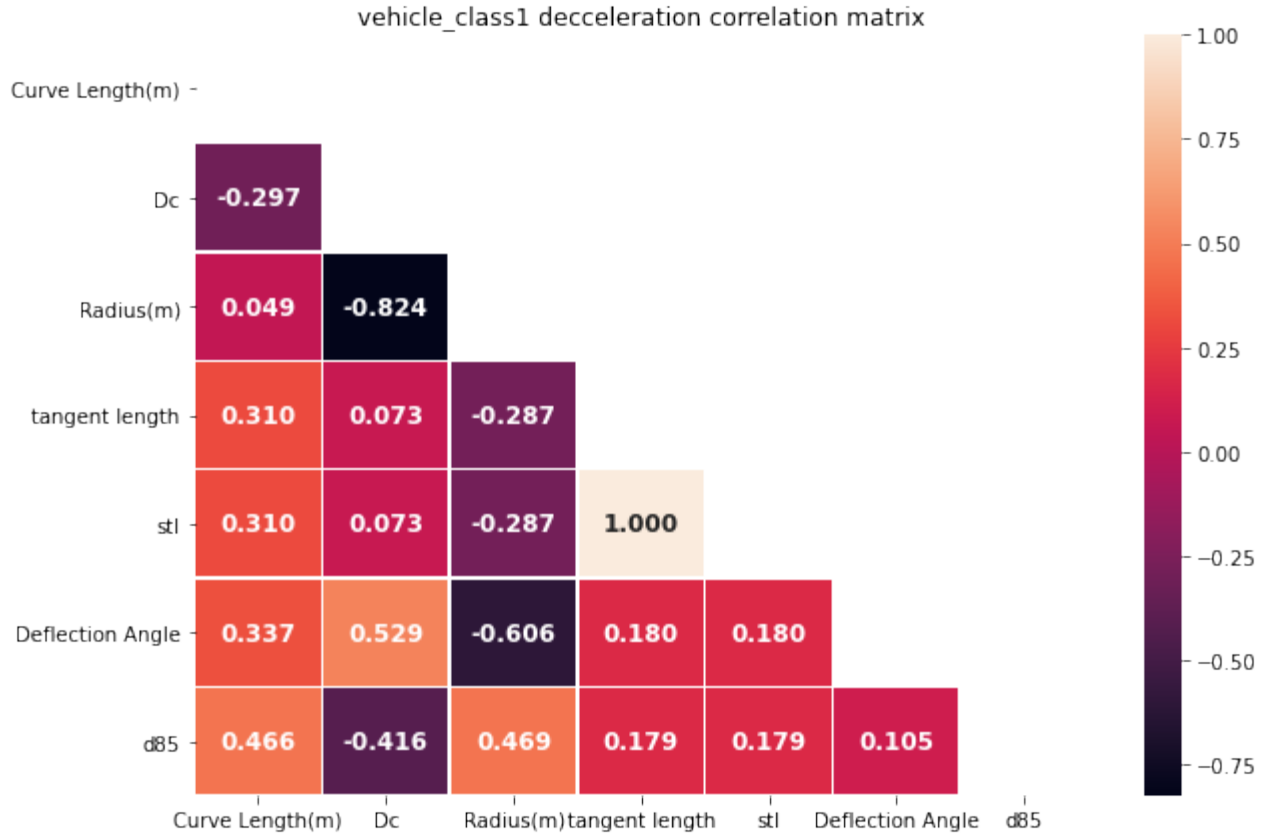
In [6]:

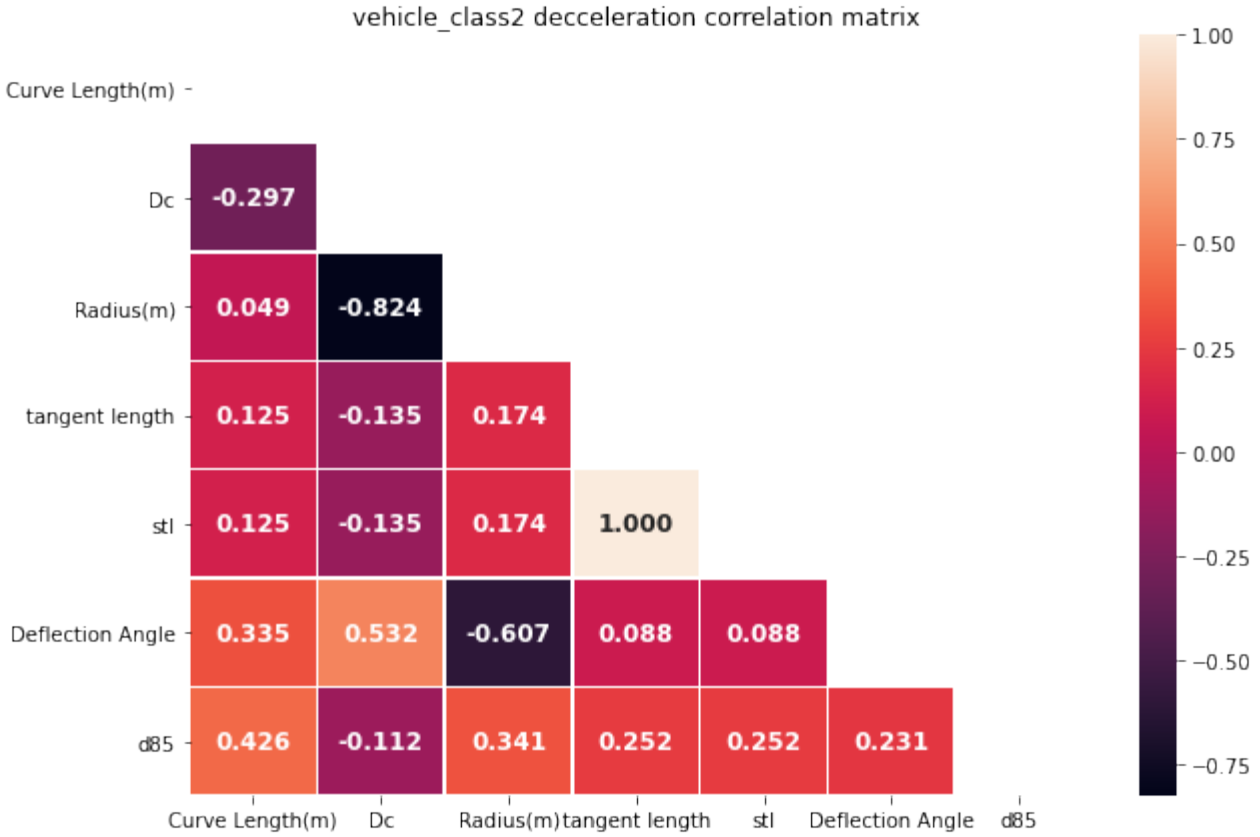
```
d85_corr= vehicle_class1[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','d85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(d85_corr.corr()))
ax=sns.heatmap(d85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
        "0.3f")
plt.title('vehicle_class1 decceleration correlation matrix')
d85_corr= vehicle_class2[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','d85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(d85_corr.corr()))
ax=sns.heatmap(d85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
        "0.3f")
plt.title('vehicle_class2 decceleration correlation matrix')
d85_corr= vehicle_class3[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','d85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(d85_corr.corr()))
ax=sns.heatmap(d85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
        "0.3f")
plt.title('vehicle_class3 decceleration correlation matrix')
d85_corr= vehicle_class4[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','d85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(d85_corr.corr()))
ax=sns.heatmap(d85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
        "0.3f")
```

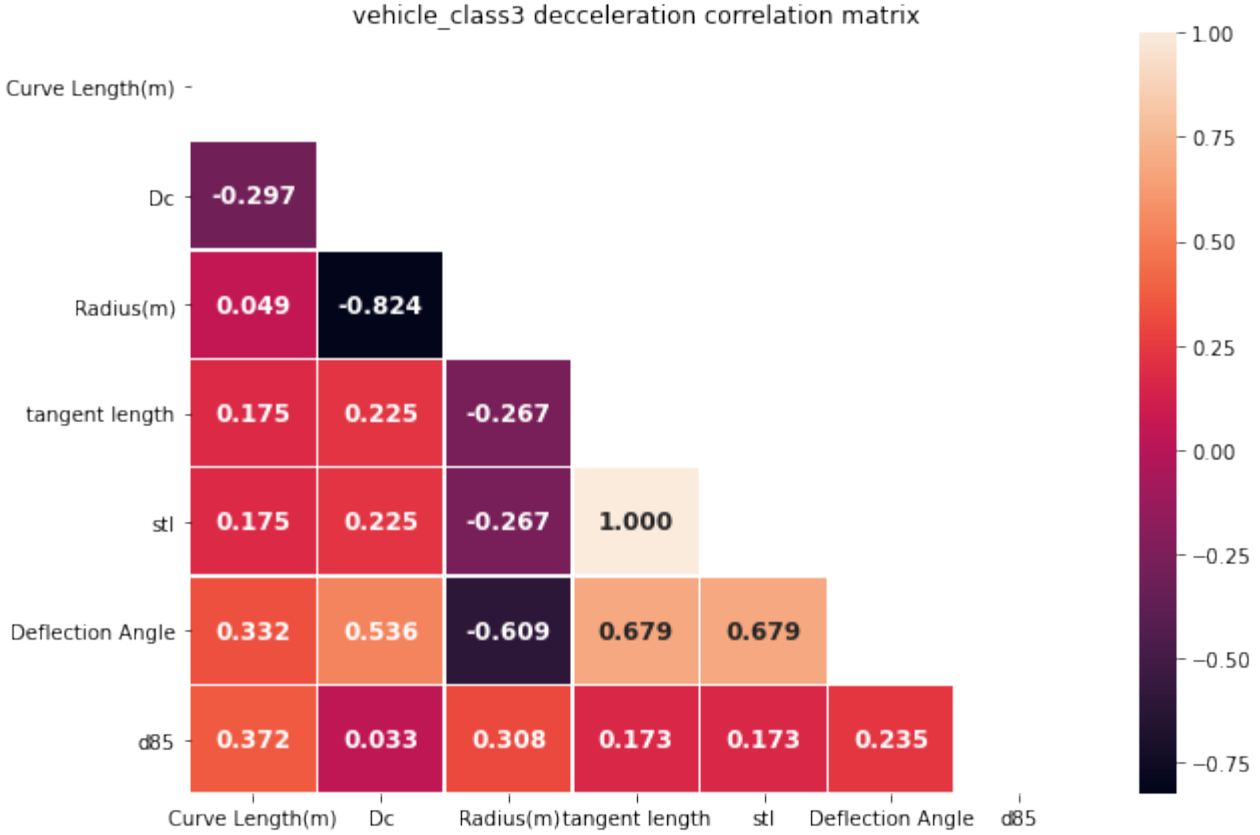
```
plt.title('vehicle_class4 decceleration correlation matrix')
d85_corr= vehicle_class5[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','d85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(d85_corr.corr()))
ax=sns.heatmap(d85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
        "0.3f")
plt.title('vehicle_class5 decceleration correlation matrix')
```

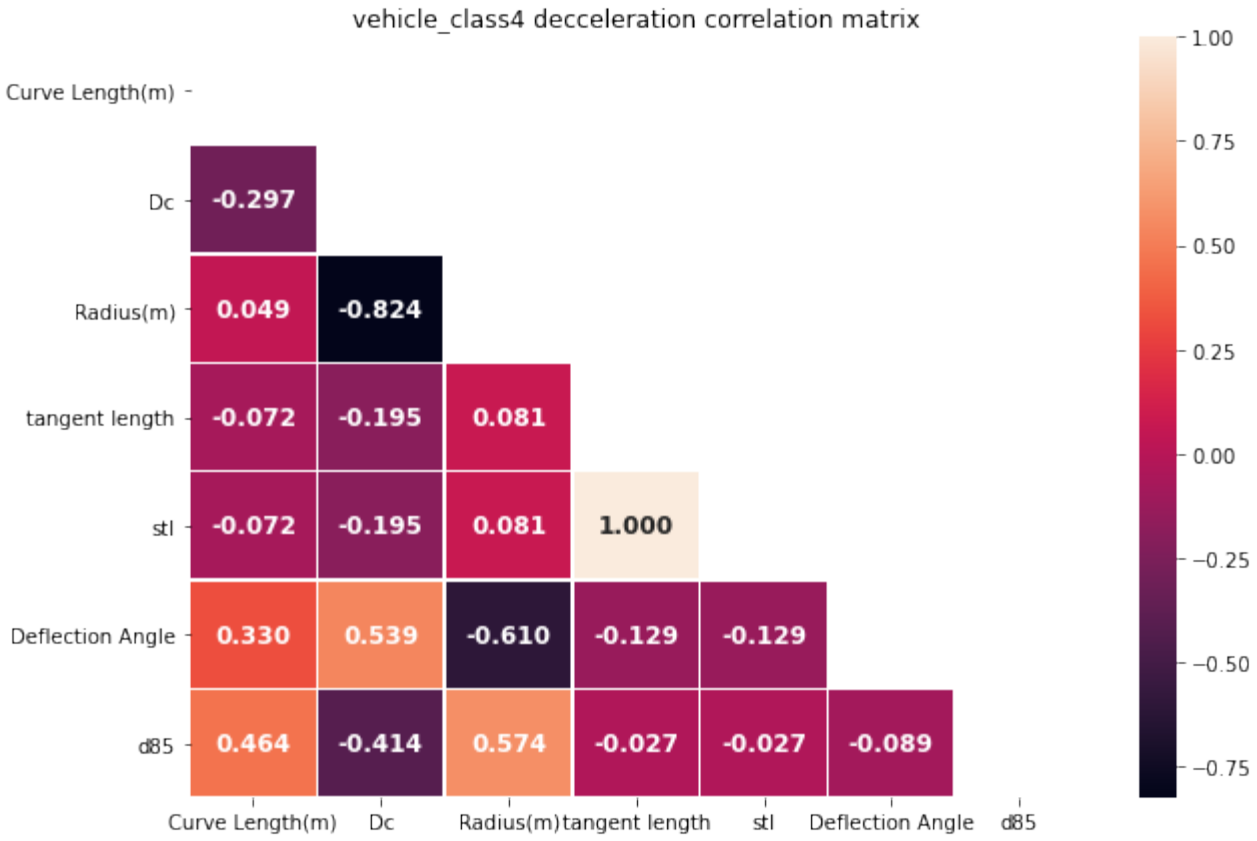
Out[6]:

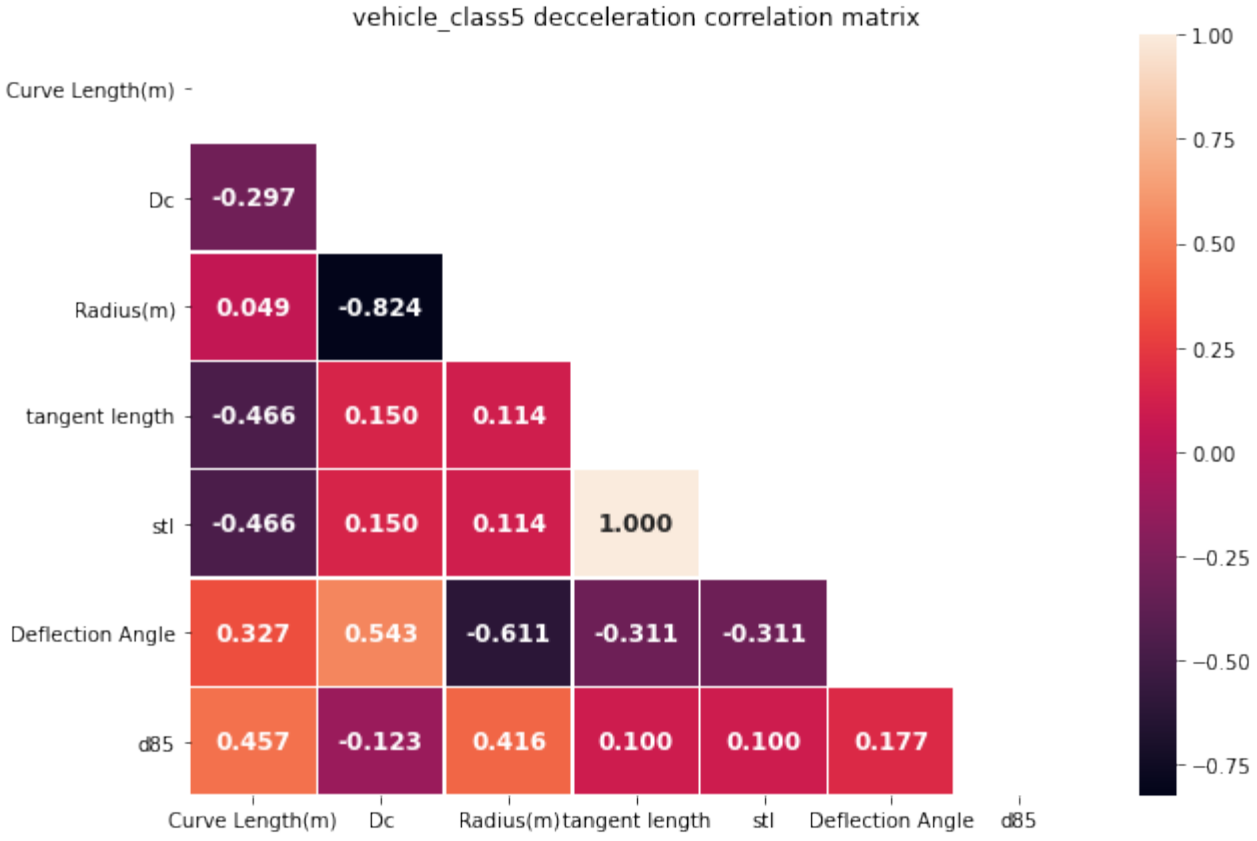
```
Text(0.5, 1.0, 'vehicle class5 decceleration correlation matrix')
```




vehicle_class2 decceleration correlation matrix
Curve Length(m)
Dc
Radius(m)
tangent length
stl
Deflection Angle
d85
-0.297
0.049 -0.824
0.125 -0.135 0.174
0.125 -0.135 0.174 1.000
0.335 0.532 -0.607 0.088 0.088
0.426 -0.112 0.341 0.252 0.252 0.231
Curve Length(m) Dc Radius(m) tangent length stl Deflection Angle d85
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


vehicle_class3 decceleration correlation matrix
Curve Length(m)
Dc
Radius(m)
tangent length
stl
Deflection Angle
d85
-0.297
0.049 -0.824
0.175 0.225 -0.267
0.175 0.225 -0.267 1.000
0.332 0.536 -0.609 0.679 0.679
0.372 0.033 0.308 0.173 0.173 0.235
Curve Length(m) Dc Radius(m) tangent length stl Deflection Angle d85
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75





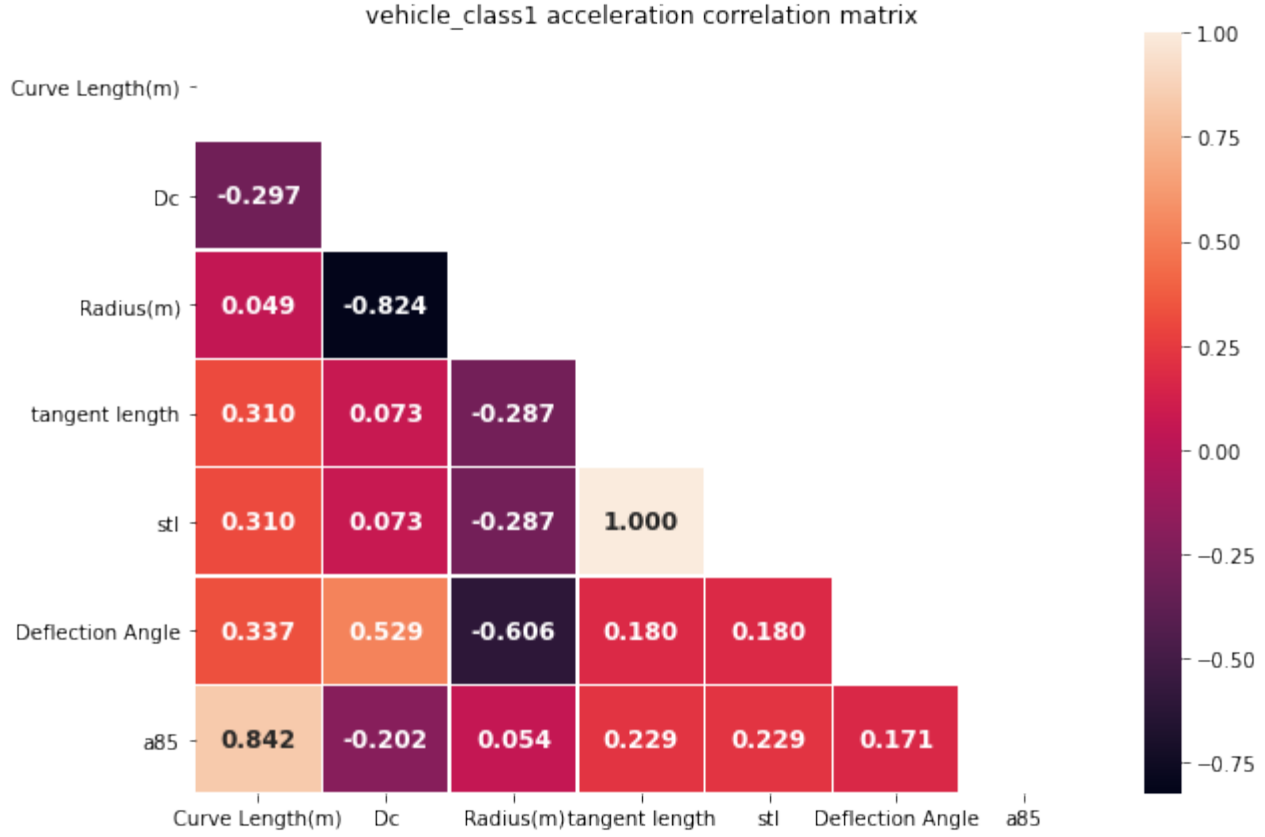
In [7]:

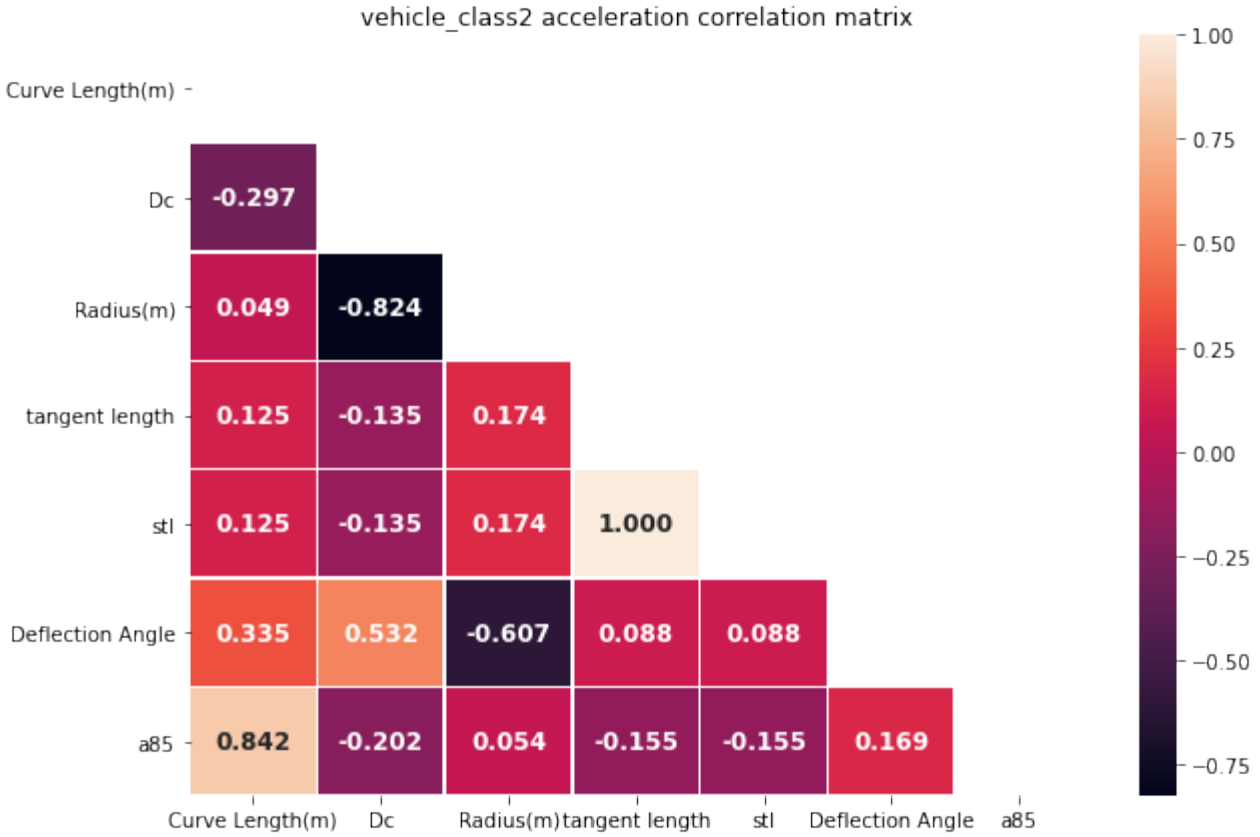
```
a85_corr= vehicle_class1[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','a85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(a85_corr.corr()))
ax=sns.heatmap(a85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
```

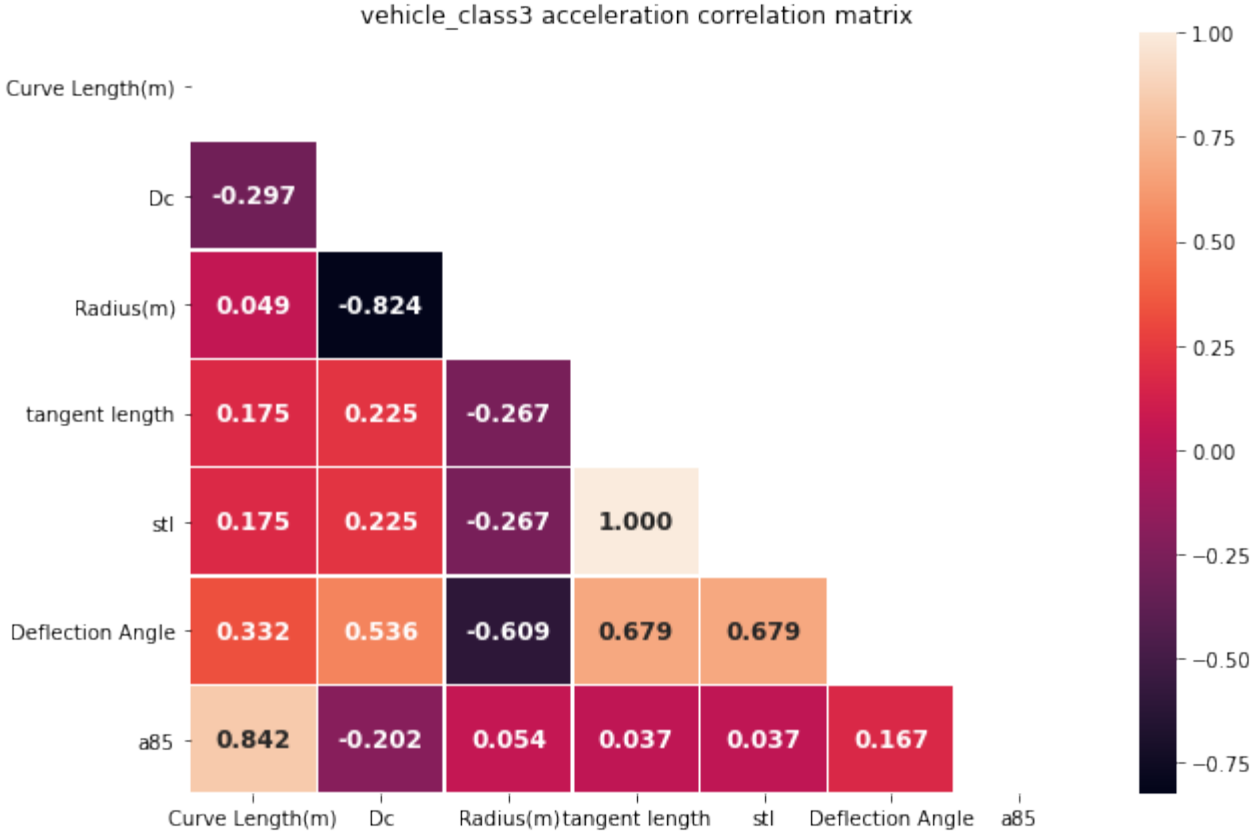
```
            "0.3f")
plt.title('vehicle_class1 acceleration correlation matrix')
a85_corr= vehicle_class2[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','a85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(a85_corr.corr()))
ax=sns.heatmap(a85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
            "0.3f")
plt.title('vehicle_class2 acceleration correlation matrix')
a85_corr= vehicle_class3[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','a85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(a85_corr.corr()))
ax=sns.heatmap(a85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
            "0.3f")
plt.title('vehicle_class3 acceleration correlation matrix')
a85_corr= vehicle_class4[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','a85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(a85_corr.corr()))
ax=sns.heatmap(a85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
            "0.3f")
plt.title('vehicle_class4 acceleration correlation matrix')
a85_corr= vehicle_class5[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','a85']]
plt.figure(figsize=(10,7))
mask= np.triu(np.ones_like(a85_corr.corr()))
ax=sns.heatmap(a85_corr.corr(), annot =True, mask= mask, annot_kws={'fontsize' : 12, 'fontweight' :'bold'},linewidths=0.5,fmt =
            "0.3f")
plt.title('vehicle_class5 acceleration correlation matrix')
```

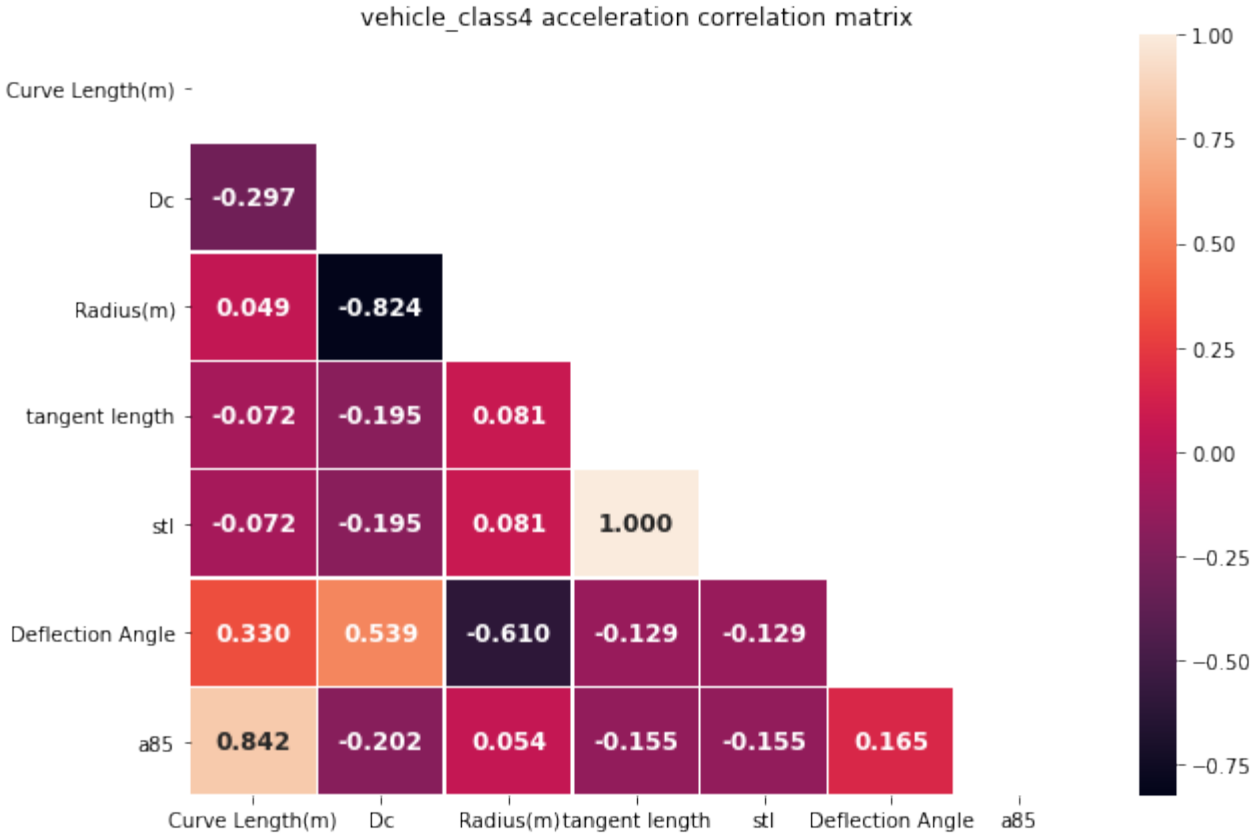
Out[7]:

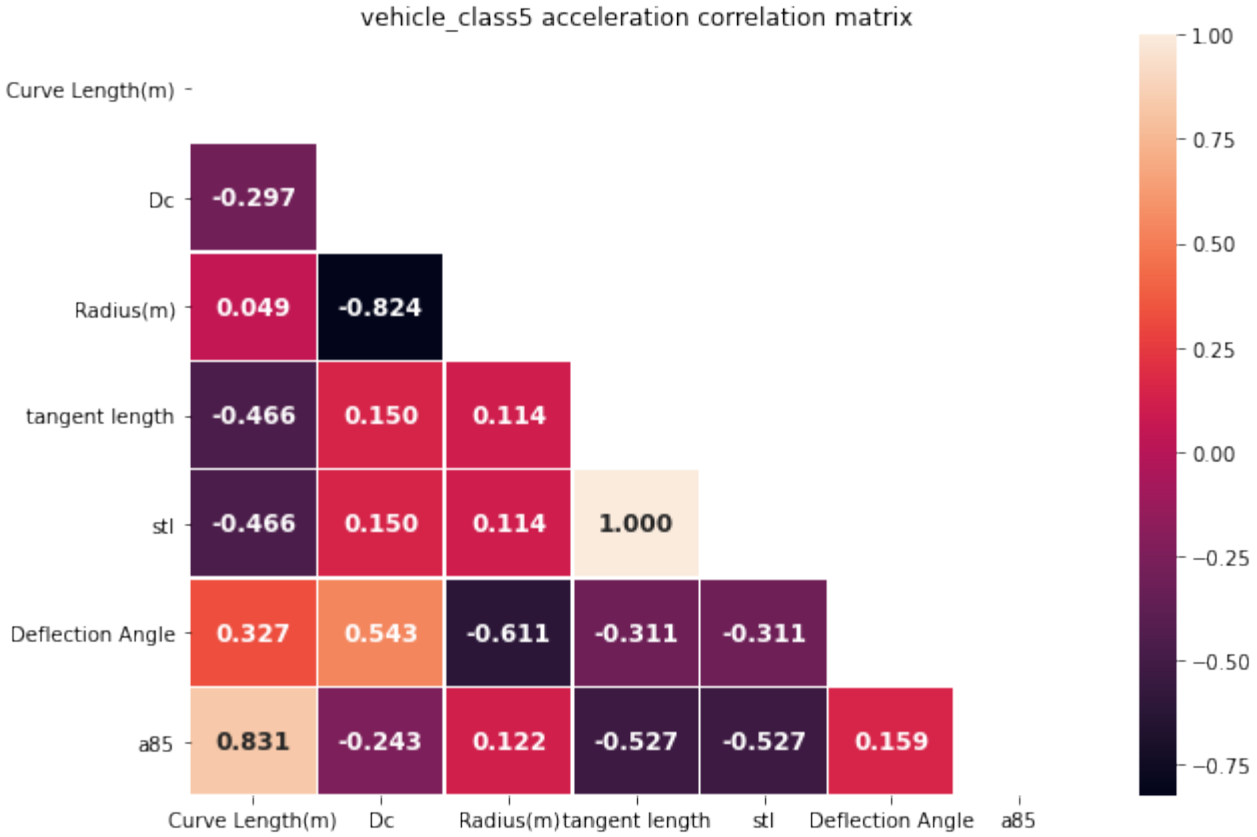
```
Text(0.5, 1.0, 'vehicle class5 acceleration correlation matrix')
```




vehicle_class2 acceleration correlation matrix
Curve Length(m)
Dc
Radius(m)
tangent length
stl
Deflection Angle
a85
-0.297
0.049 -0.824
0.125 -0.135 0.174
0.125 -0.135 0.174 1.000
0.335 0.532 -0.607 0.088 0.088
0.842 -0.202 0.054 -0.155 -0.155 0.169
Curve Length(m) Dc Radius(m) tangent length stl Deflection Angle a85
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


vehicle_class3 acceleration correlation matrix
Curve Length(m)
Dc
Radius(m)
tangent length
stl
Deflection Angle
a85
-0.297
0.049 -0.824
0.175 0.225 -0.267
0.175 0.225 -0.267 1.000
0.332 0.536 -0.609 0.679 0.679
0.842 -0.202 0.054 0.037 0.037 0.167
Curve Length(m) Dc Radius(m) tangent length stl Deflection Angle a85
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75





In [8]:

```
y= vehicle_class1['d85']
x= vehicle_class1[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','∆v85']]

#splitting data into training and test split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)

poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)
m= sm.add_constant(x_train_df)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()

y= vehicle_class2['d85']
x= vehicle_class2[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)

poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)
m= sm.add_constant(x_train_df)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()

y= vehicle_class3['d85']
x= vehicle_class3[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)

poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)
m= sm.add_constant(x_train_df)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()

y= vehicle_class4['d85']
x= vehicle_class4[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)

poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)
m= sm.add_constant(x_train_df)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
```

```
results.summary()

y= vehicle_class5['d85']
x= vehicle_class5[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)

poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)
m= sm.add_constant(x_train_df)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[8]:

OLS Regression Results

Dep. Variable:	d85	**R-squared:**	0.986
Model:	OLS	**Adj. R-squared:**	0.945
Method:	Least Squares	**F-statistic:**	24.12
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.0403
Time:	00:37:26	**Log-Likelihood:**	22.574
No. Observations:	9	**AIC:**	-31.15
Df Residuals:	2	**BIC:**	-29.77
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.7536	0.421	-1.790	0.215	-2.565	1.057
Curve Length(m)	0.0016	0.001	1.897	0.198	-0.002	0.005
Dc	0.0115	0.008	1.503	0.272	-0.021	0.044
Radius(m)	0.0004	0.000	4.224	0.052	-8.15e-06	0.001
tangent length	8.344e-05	0.000	0.383	0.739	-0.001	0.001

stl	8.344e-05	0.000	0.383	0.739	-0.001	0.001
Deflection Angle	-0.0047	0.008	-0.557	0.633	-0.041	0.031
Δv85	-0.0131	0.010	-1.327	0.316	-0.055	0.029

Omnibus:	1.762	**Durbin-Watson:**	2.347
Prob(Omnibus):	0.414	**Jarque-Bera (JB):**	0.766
Skew:	-0.126	**Prob(JB):**	0.682
Kurtosis:	1.593	**Cond. No.**	2.76e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.26e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [9]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[9]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	44.595465	94.272539	21.310276	inf	inf	82.354322	3.545481

In [10]:

```
y= data['a85']
x= data[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 15)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	a85	**R-squared:**	0.636
Model:	OLS	**Adj. R-squared:**	0.579
Method:	Least Squares	**F-statistic:**	11.07

Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	4.08e-07
Time:	00:37:27	**Log-Likelihood:**	151.31
No. Observations:	45	**AIC:**	-288.6
Df Residuals:	38	**BIC:**	-276.0
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0029	0.010	0.292	0.772	-0.017	0.023
Curve Length(m)	7.722e-05	1.18e-05	6.561	0.000	5.34e-05	0.000
Dc	0.0002	0.000	1.108	0.275	-0.000	0.000
Radius(m)	1.661e-06	6.57e-06	0.253	0.802	-1.16e-05	1.5e-05
tangent length	-1.361e-05	4.82e-06	-2.825	0.007	-2.34e-05	-3.86e-06
stl	-1.361e-05	4.82e-06	-2.825	0.007	-2.34e-05	-3.86e-06
Deflection Angle	-0.0001	0.000	-1.219	0.230	-0.000	8.93e-05
Δv85	4.301e-05	0.000	0.207	0.837	-0.000	0.000

Omnibus:	19.951	**Durbin-Watson:**	1.756
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	61.623
Skew:	0.907	**Prob(JB):**	4.16e-14
Kurtosis:	8.438	**Cond. No.**	4.22e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.63e-26. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [11]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[11]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	6.009703	7.515231	4.44921	inf	inf	21.396011	2.017927

In [12]:

```
#vehicle_class1 model
y=vehicle_class1['v85']
x= vehicle_class1[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','∆v85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[12]:

OLS Regression Results

Dep. Variable:	v85	**R-squared:**	0.936
Model:	OLS	**Adj. R-squared:**	0.743
Method:	Least Squares	**F-statistic:**	4.851
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.181
Time:	00:37:27	**Log-Likelihood:**	-20.116
No. Observations:	9	**AIC:**	54.23
Df Residuals:	2	**BIC:**	55.61
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	44.7225	13.728	3.258	0.083	-14.346	103.791
Curve Length(m)	0.0014	0.031	0.046	0.968	-0.133	0.136
Dc	-1.1648	0.280	-4.163	0.053	-2.369	0.039
Radius(m)	-0.0173	0.011	-1.630	0.245	-0.063	0.028
tangent length	0.0407	0.009	4.427	0.047	0.001	0.080
stl	0.0407	0.009	4.427	0.047	0.001	0.080
Deflection Angle	0.3408	0.241	1.412	0.293	-0.698	1.379
∆v85	1.4828	0.400	3.711	0.066	-0.236	3.202

Omnibus:	1.417	**Durbin-Watson:**	1.832
Prob(Omnibus):	0.492	**Jarque-Bera (JB):**	0.893
Skew:	-0.485	**Prob(JB):**	0.640
Kurtosis:	1.800	**Cond. No.**	1.15e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.55e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [13]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[13]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	17.503336	33.199761	12.797775	inf	inf	56.491664	6.14844

In [14]:

```
#vehicle_class2 model
y=vehicle_class2['v85']
x= vehicle_class2[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[14]:

OLS Regression Results

Dep. Variable:	v85	**R-squared:**	0.610
Model:	OLS	**Adj. R-squared:**	-0.560
Method:	Least Squares	**F-statistic:**	0.5214
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.773
Time:	00:37:28	**Log-Likelihood:**	-25.987
No. Observations:	9	**AIC:**	65.97
Df Residuals:	2	**BIC:**	67.36
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	80.8922	34.911	2.317	0.146	-69.319	231.103
Curve Length(m)	0.1216	0.137	0.888	0.468	-0.468	0.711
Dc	0.0131	0.509	0.026	0.982	-2.177	2.203
Radius(m)	0.0095	0.023	0.407	0.724	-0.091	0.110
tangent length	-0.0475	0.036	-1.302	0.323	-0.204	0.109
stl	-0.0475	0.036	-1.302	0.323	-0.204	0.109
Deflection Angle	0.2519	0.568	0.444	0.701	-2.191	2.695
Δv85	0.2005	0.501	0.400	0.728	-1.954	2.355

Omnibus:	4.439	**Durbin-Watson:**	2.470
Prob(Omnibus):	0.109	**Jarque-Bera (JB):**	1.194
Skew:	-0.307	**Prob(JB):**	0.550
Kurtosis:	1.324	**Cond. No.**	1.38e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.93e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [15]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[15]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	80.548631	29.54392	9.192605	inf	inf	47.368387	1.666123

In [16]:

```
#vehicle_class3 model
y=vehicle_class3['v85']
x= vehicle_class3[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
```

```
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[16]:

OLS Regression Results

Dep. Variable:	v85	**R-squared:**	0.381
Model:	OLS	**Adj. R-squared:**	-1.475
Method:	Least Squares	**F-statistic:**	0.2054
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.945
Time:	00:37:28	**Log-Likelihood:**	-24.782
No. Observations:	9	**AIC:**	63.56
Df Residuals:	2	**BIC:**	64.94
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	66.4565	24.412	2.722	0.113	-38.580	171.493
Curve Length(m)	-0.0109	0.077	-0.141	0.901	-0.342	0.321
Dc	-0.1932	0.481	-0.402	0.727	-2.264	1.877
Radius(m)	-0.0061	0.018	-0.335	0.769	-0.085	0.073
tangent length	0.0087	0.013	0.669	0.572	-0.047	0.065
stl	0.0087	0.013	0.669	0.572	-0.047	0.065
Deflection Angle	-0.0500	0.746	-0.067	0.953	-3.262	3.162
Δv85	-0.0367	0.786	-0.047	0.967	-3.420	3.347

Omnibus:	3.986	**Durbin-Watson:**	2.895
Prob(Omnibus):	0.136	**Jarque-Bera (JB):**	1.430
Skew:	-0.972	**Prob(JB):**	0.489
Kurtosis:	3.187	**Cond. No.**	2.83e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.06e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [17]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
```

```
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[17]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	31.936819	34.134388	4.193424	inf	inf	151.200052	3.714119

In [18]:

```
#vehicle_class4 model
y=vehicle_class4['v85']
x= vehicle_class4[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[18]:

OLS Regression Results

Dep. Variable:	v85	**R-squared:**	0.979
Model:	OLS	**Adj. R-squared:**	0.916
Method:	Least Squares	**F-statistic:**	15.51
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.0618
Time:	00:37:28	**Log-Likelihood:**	-16.895
No. Observations:	9	**AIC:**	47.79
Df Residuals:	2	**BIC:**	49.17
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-85.0796	17.345	-4.905	0.039	-159.711	-10.449
Curve Length(m)	0.0953	0.023	4.236	0.051	-0.001	0.192
Dc	0.8718	0.156	5.598	0.030	0.202	1.542
Radius(m)	0.0681	0.008	8.469	0.014	0.033	0.103
tangent length	0.0210	0.004	5.041	0.037	0.003	0.039

stl	0.0210	0.004	5.041	0.037	0.003	0.039
Deflection Angle	0.7088	0.185	3.822	0.062	-0.089	1.507
Δv85	2.2731	0.401	5.663	0.030	0.546	4.000

Omnibus:	1.042	**Durbin-Watson:**	1.952
Prob(Omnibus):	0.594	**Jarque-Bera (JB):**	0.686
Skew:	0.292	**Prob(JB):**	0.710
Kurtosis:	1.780	**Cond. No.**	2.74e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.28e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [19]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[19]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85
vif	16.79488	18.145347	5.892486	inf	inf	55.966701	2.501265

In [20]:

```
#vehicle_class5 model
y=vehicle_class5['v85']
x= vehicle_class5[['Curve Length(m)','Dc','Radius(m)','tangent length','stl','Deflection Angle','Δv85']]

#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
m= sm.add_constant(x_train)
model= sm.OLS(y_train, m)
results= model.fit()
results.params
results.summary()
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurto
sistest only valid for n>=20 ... continuing anyway, n=9
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[20]:

OLS Regression Results

Dep. Variable:	v85	**R-squared:**	0.574

Model:	OLS	**Adj. R-squared:**	-0.704
Method:	Least Squares	**F-statistic:**	0.4492
Date:	Sun, 02 Apr 2023	**Prob (F-statistic):**	0.811
Time:	00:37:29	**Log-Likelihood:**	-27.034
No. Observations:	9	**AIC:**	68.07
Df Residuals:	2	**BIC:**	69.45
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	41.7828	104.240	0.401	0.727	-406.726	490.292
Curve Length(m)	-0.0316	0.203	-0.156	0.890	-0.904	0.840
Dc	-0.2992	1.890	-0.158	0.889	-8.432	7.833
Radius(m)	0.0184	0.026	0.718	0.548	-0.092	0.128
tangent length	0.0069	0.054	0.127	0.910	-0.225	0.239
stl	0.0069	0.054	0.127	0.910	-0.225	0.239
Deflection Angle	0.4695	2.075	0.226	0.842	-8.460	9.399
Δv85	0.4161	2.441	0.170	0.880	-10.088	10.920

Omnibus:	2.441	**Durbin-Watson:**	2.437
Prob(Omnibus):	0.295	**Jarque-Bera (JB):**	0.864
Skew:	0.102	**Prob(JB):**	0.649
Kurtosis:	1.496	**Cond. No.**	2.76e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.26e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [21]:

```
#residual checks
x_test = sm.add_constant(x_test)
y_pred= results.predict(x_test)
residual= y_test-y_pred
vif=[variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
pd.DataFrame({'vif': vif[0:]}, index=x_train.columns).T
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193:
RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r squared i)
```

Out[21]:

	Curve Length(m)	Dc	Radius(m)	tangent length	stl	Deflection Angle	Δv85

vif	44.595465	94.272539	21.310276	inf	inf	82.354322	3.545481

In [22]:

```
density= 2/data['Lane Width(m)']
data['density']= density
data.head(5)
```

Out[22]:

	Vehicle	AT	BC	centre	MP	EC	DT	Radius(m)	Dc	Lane Width(m)	...	acc rate	tangent length	stl	Δv85	den
0	Class 1	92.700	58.16	62.680000	55.46	74.42	85.5	100	57.272727	3.50	...	0.0	300	300	25.517000	0.57
1	Class 1	102.400	101.50	96.266667	93.40	93.90	98.4	119	48.128342	3.50	...	0.0	700	700	13.033333	0.57
2	Class 1	85.585	84.00	68.566667	60.50	61.20	69.8	172	33.298097	3.25	...	0.0	500	500	0.311667	0.61
3	Class 1	100.400	100.00	83.433333	74.00	76.30	90.3	220	26.033058	3.50	...	0.0	750	750	12.129500	0.57
4	Class 1	95.610	94.71	88.576667	84.91	86.11	97.1	297	19.283747	3.25	...	0.0	600	600	-9.231000	0.61

5 rows × 24 columns

In [23]:

```
#splitting data into training and test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3, random_state= 10)
x_train_df,x_test_df= pd.DataFrame(x_train), pd.DataFrame(x_test)
from sklearn import linear_model
poly= PolynomialFeatures(degree=3, include_bias=False)
x_train_poly,x_test_poly= poly.fit_transform(x_train_df), poly.fit_transform(x_test_df)

model=linear_model.LinearRegression()
model= model.fit(x_train_poly,y_train)
coefficient= model.coef_
intercept= model.intercept_
```

In [24]:

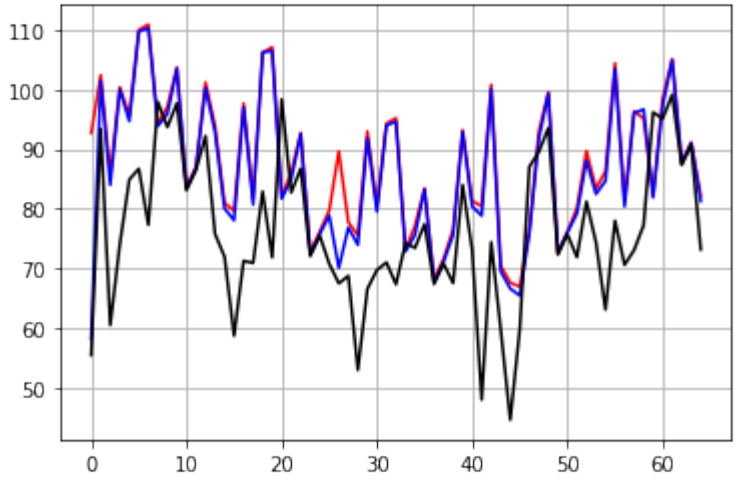
```
from sklearn.metrics import r2_score
prediction= model.predict(x_test_poly)
r2_score(prediction,y_test)
```

Out[24]:

```
-0.12402161679779389
```

In [25]:

```
plt.figsize=(10,7)
x= data['AT']
y=data['BC']
z=data['MP']
plt.plot(x,'r',y,'b',z,'k')
plt.grid()
```

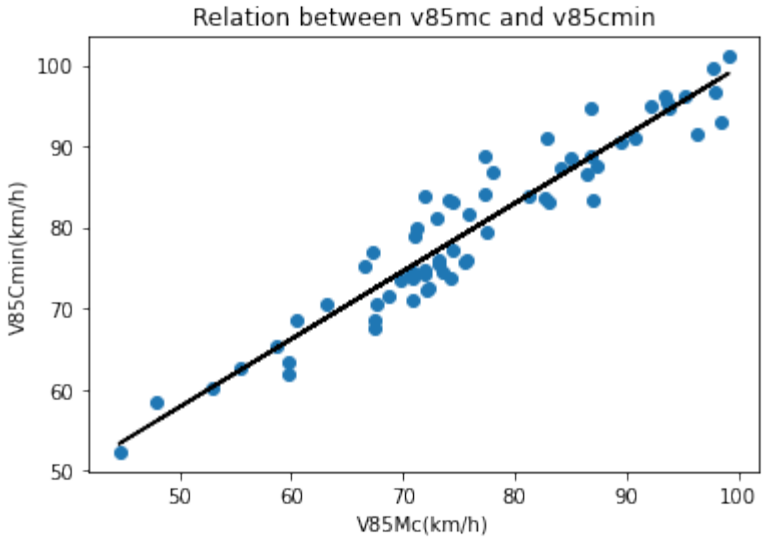


In [26]:

```
y=data.centre
x=data.MP
plt.figsize=(10,7)
plt.scatter(x,y,label='Relation between v85mc and v85cmin')
plt.ylabel('V85Cmin(km/h)')
plt.xlabel('V85Mc(km/h)')
n, b= np.polyfit(x,y,1)
plt.plot(x,n*x + b, color='k')
plt.title('Relation between v85mc and v85cmin')
```

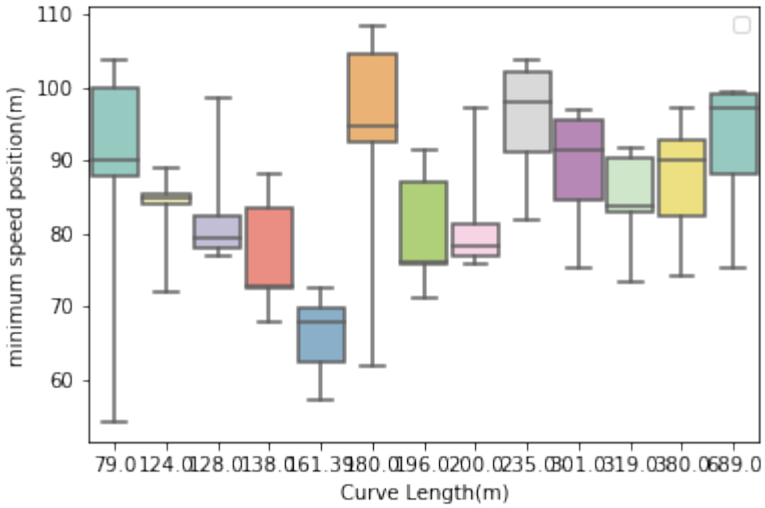
Out[26]:

```
Text(0.5, 1.0, 'Relation between v85mc and v85cmin')
```



In [27]:

```
plt.figsize=(20,10)
whis_perc=[0,100]
ax=sns.boxplot(x='Curve
Length(m)',y='DT',data=data,width=0.9,orient='v',palette='Set3',whis=whis_perc)
handles,labels=ax.get_legend_handles_labels()
labels=[f"{perc}th percentile" for perc in whis_perc]
ax.legend(handles,labels)
plt.ylabel('minimum speed position(m)')
plt.show()
```

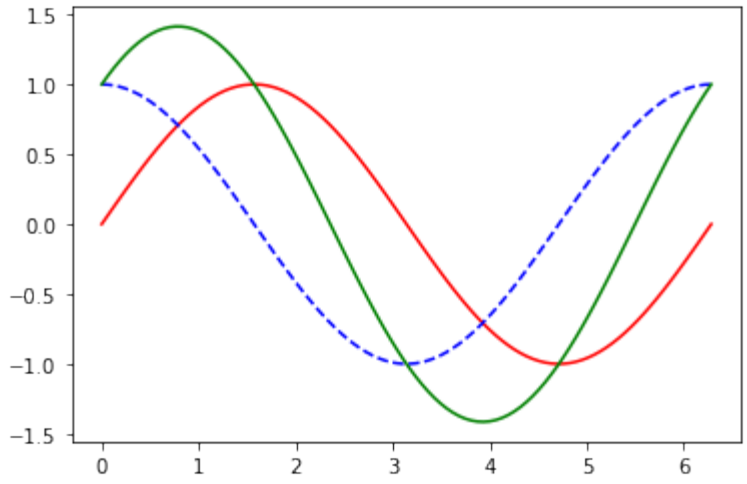


In [28]:

```
x=np.linspace(0,2*np.pi,100)
f=lambda x:np.sin(x)
g=lambda x:np.cos(x)
h=lambda x:f(x) + g(x)
fig, ax =plt.subplots()
ax.plot(x,f(x),color='r',linestyle='-')
ax.plot(x,g(x),color='b',linestyle='--')
ax.plot(x,h(x),color='g')
```

Out[28]:

```
[<matplotlib.lines.Line2D at 0x1cebd9f6580>]
```



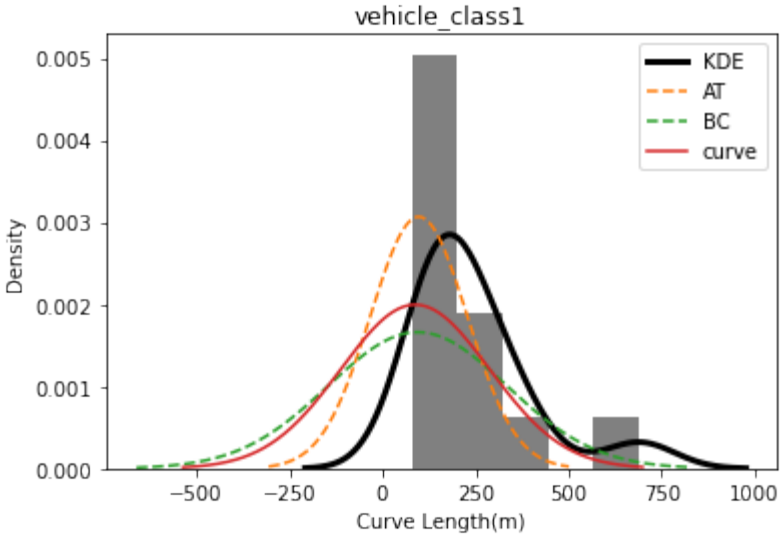
In [29]:

```
sns.distplot(x=vehicle_class1['Curve
Length(m)'],kde=True,rug=False,bins=5,kde_kws={"color": "k", "lw": 3, "label": "KDE"},
                        hist_kws={"linewidth": 3,
                                  "alpha": 1, "color": "grey"})
sns.kdeplot(data=vehicle_class1,x='AT',bw_adjust=25,linestyle='--',label='AT')
sns.kdeplot(data=vehicle_class1,x='BC',bw_adjust=29,linestyle='dashed',label='BC')
sns.kdeplot(data=vehicle_class1,x='centre',bw_adjust=30,label='curve')
plt.xlabel('Curve Length(m)')
plt.legend()
plt.title('vehicle_class1')
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
  `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]:

```
Text(0.5, 1.0, 'vehicle class1')
```



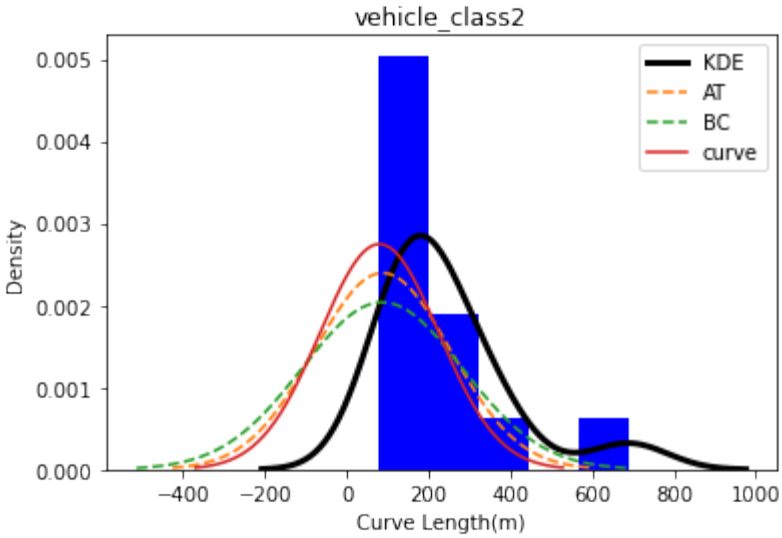
In [41]:

```
sns.distplot(x=vehicle_class2['Curve
Length(m)'],kde=True,rug=False,bins=5,kde_kws={"color": "k", "lw": 3, "label": "KDE"},
                        hist_kws={"linewidth": 3,
                                  "alpha": 1, "color": "blue"})
sns.kdeplot(data=vehicle_class2,x='AT',bw_adjust=25,linestyle='--',label='AT')
sns.kdeplot(data=vehicle_class2,x='BC',bw_adjust=29,linestyle='dashed',label='BC')
sns.kdeplot(data=vehicle_class2,x='centre',bw_adjust=30,label='curve')
plt.xlabel('Curve Length(m)')
plt.legend()
plt.title('vehicle_class2')
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
  `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[41]:

```
Text(0.5, 1.0, 'vehicle class2')
```



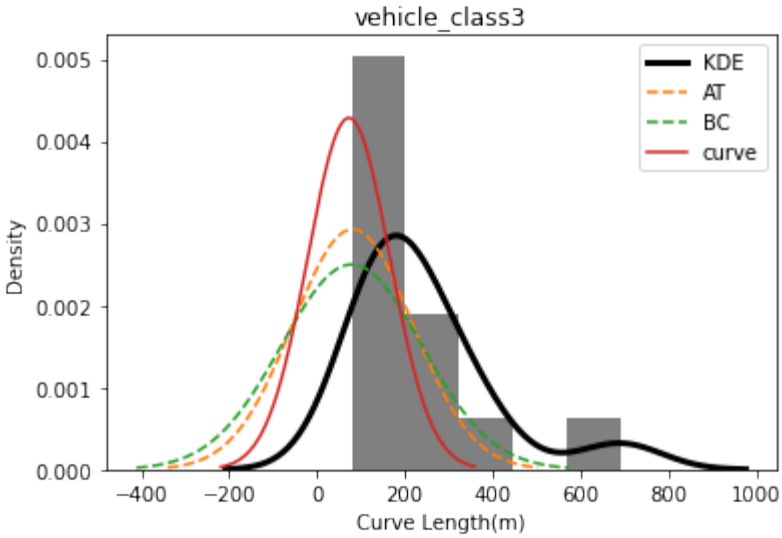
In [31]:

```
sns.distplot(x=vehicle_class3['Curve
Length(m)'],kde=True,rug=False,bins=5,kde_kws={"color": "k", "lw": 3, "label": "KDE"},
                        hist_kws={"linewidth": 3,
                                  "alpha": 1, "color": "grey"})
sns.kdeplot(data=vehicle_class3,x='AT',bw_adjust=25,linestyle='--',label='AT')
sns.kdeplot(data=vehicle_class3,x='BC',bw_adjust=29,linestyle='dashed',label='BC')
sns.kdeplot(data=vehicle_class3,x='centre',bw_adjust=30,label='curve')
plt.xlabel('Curve Length(m)')
plt.legend()
plt.title('vehicle_class3')
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
 `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[31]:

```
Text(0.5, 1.0, 'vehicle class3')
```



In [32]:

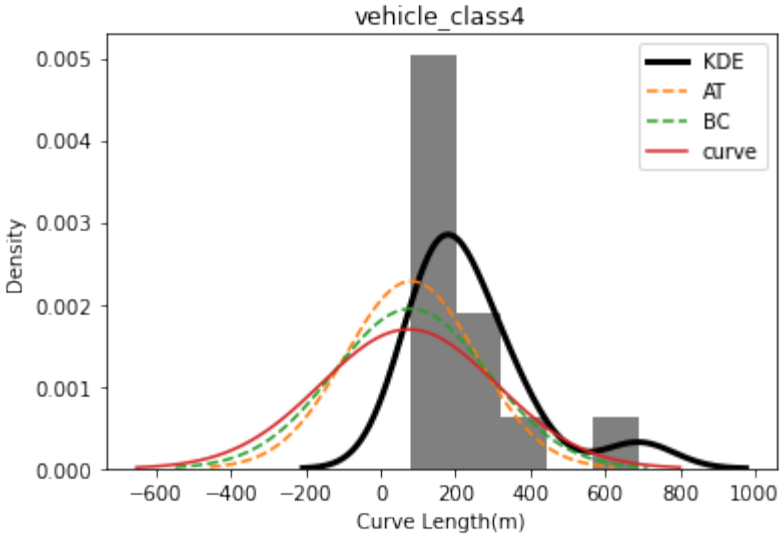
```
sns.distplot(x=vehicle_class4['Curve
Length(m)'],kde=True,rug=False,bins=5,kde_kws={"color": "k", "lw": 3, "label": "KDE"},
                        hist_kws={"linewidth": 3,
```

```
                                    "alpha": 1, "color": "grey"})
sns.kdeplot(data=vehicle_class4,x='AT',bw_adjust=25,linestyle='--',label='AT')
sns.kdeplot(data=vehicle_class4,x='BC',bw_adjust=29,linestyle='dashed',label='BC')
sns.kdeplot(data=vehicle_class4,x='centre',bw_adjust=30,label='curve')
plt.xlabel('Curve Length(m)')
plt.legend()
plt.title('vehicle_class4')
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
 `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[32]:

```
Text(0.5, 1.0, 'vehicle_class4')
```



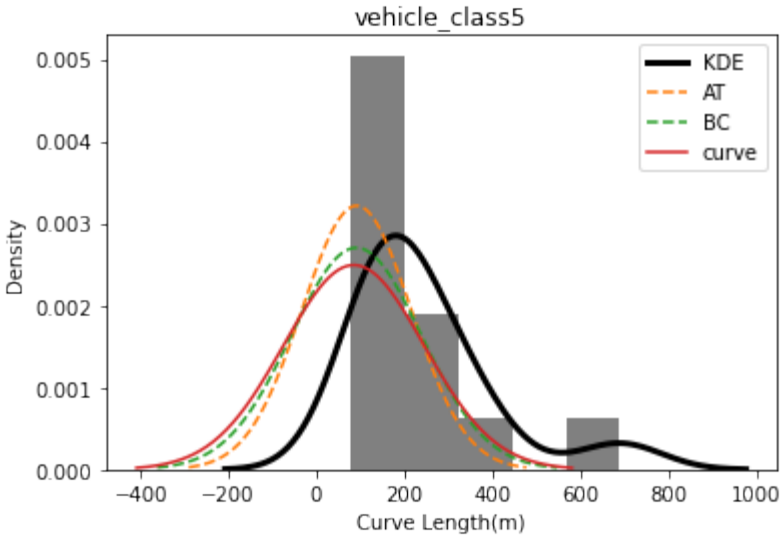
In [33]:

```
sns.distplot(x=vehicle_class5['Curve
Length(m)'],kde=True,rug=False,bins=5,kde_kws={"color": "k", "lw": 3, "label": "KDE"},
                        hist_kws={"linewidth": 3,
                                  "alpha": 1, "color": "grey"})
sns.kdeplot(data=vehicle_class5,x='AT',bw_adjust=25,linestyle='--',label='AT')
sns.kdeplot(data=vehicle_class5,x='BC',bw_adjust=29,linestyle='dashed',label='BC')
sns.kdeplot(data=vehicle_class5,x='centre',bw_adjust=30,label='curve')
plt.xlabel('Curve Length(m)')
plt.legend()
plt.title('vehicle_class5')
```

```
C:\Users\LEGENDARY\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
 `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[33]:

```
Text(0.5, 1.0, 'vehicle_class5')
```

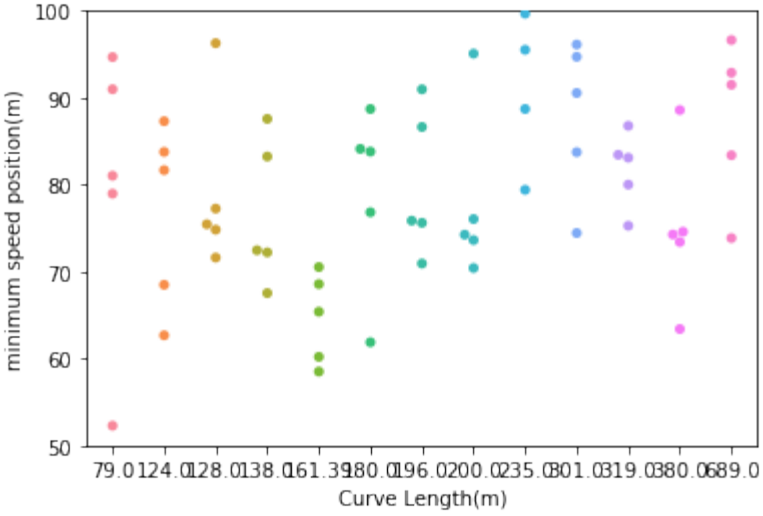


In [34]:

```
plt.xlim(50,700)
plt.ylim(50,100)
sns.swarmplot(x=data['Curve Length(m)'], y=data.centre)
plt.ylabel('minimum speed position(m)')
```

Out[34]:

```
Text(0, 0.5, 'minimum speed position(m)')
```



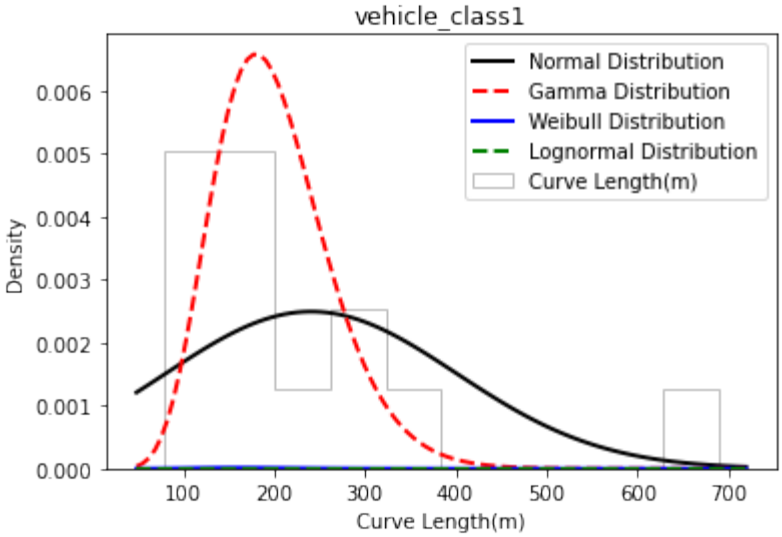
In [35]:

```
from scipy.stats import norm, gamma, weibull_min, lognorm
x=vehicle_class1['Curve Length(m)']
plt.hist(x,bins=10,density=True, alpha=0.5, label='Curve
Length(m)',histtype='step',color='grey')
xmin,xmax= plt.xlim()
y=np.linspace(xmin,xmax,100)
p= norm.pdf(y, loc=x.mean(),scale=x.std())
plt.plot(y,p,'k',linewidth=2,label='Normal Distribution')


a,loc,scale=10,0,20
p=gamma.pdf(y,a,loc=loc,scale=scale)
plt.plot(y,p,'r',linewidth=2,label='Gamma Distribution',linestyle='--')
```

```
c,loc,scale=10,0,20
p=weibull_min.pdf(y,p,loc=loc, scale=scale)
plt.plot(y,p,'b',linewidth=2,label='Weibull Distribution')

d,loc,scale=10,0,20
p=lognorm.pdf(y,p,loc=loc,scale=scale)
plt.plot(y,p,'g',linewidth=2,label='Lognormal Distribution',linestyle='dashed')
plt.legend()
plt.ylabel('Density')
plt.xlabel('Curve Length(m)')
plt.title('vehicle_class1')
```

Out[35]:

```
Text(0.5, 1.0, 'vehicle class1')
```

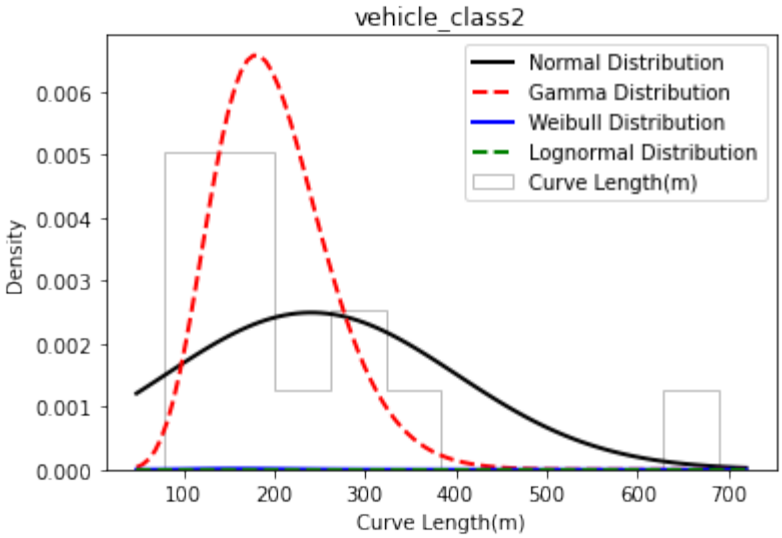


In [36]:

```
x=vehicle_class2['Curve Length(m)']
plt.hist(x,bins=10,density=True, alpha=0.5, label='Curve
Length(m)',histtype='step',color='grey')
xmin,xmax= plt.xlim()
y=np.linspace(xmin,xmax,100)
p= norm.pdf(y, loc=x.mean(),scale=x.std())
plt.plot(y,p,'k',linewidth=2,label='Normal Distribution')


a,loc,scale=10,0,20
p=gamma.pdf(y,a,loc=loc,scale=scale)
plt.plot(y,p,'r',linewidth=2,label='Gamma Distribution',linestyle='--')

c,loc,scale=10,0,20
p=weibull_min.pdf(y,p,loc=loc, scale=scale)
plt.plot(y,p,'b',linewidth=2,label='Weibull Distribution')

d,loc,scale=10,0,20
p=lognorm.pdf(y,p,loc=loc,scale=scale)
plt.plot(y,p,'g',linewidth=2,label='Lognormal Distribution',linestyle='dashed')
plt.legend()
plt.ylabel('Density')
plt.xlabel('Curve Length(m)')
plt.title('vehicle_class2')
```

Out[36]:

```
Text(0.5, 1.0, 'vehicle class2')
```

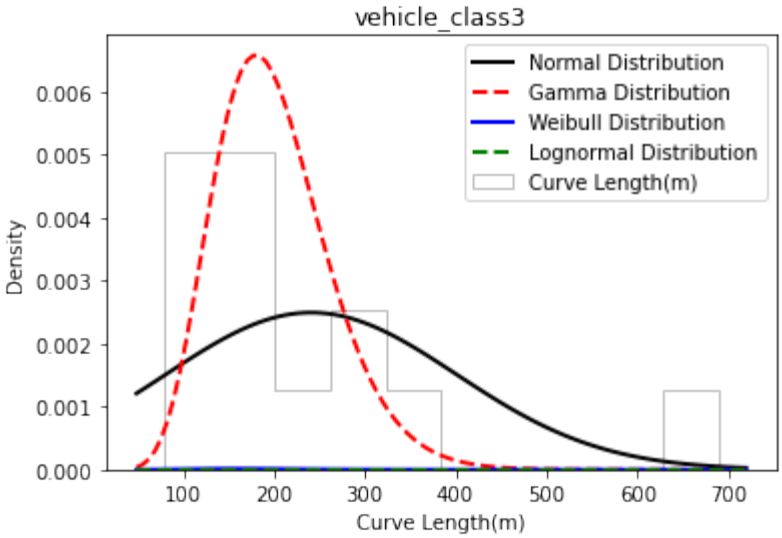


In [37]:

```
x=vehicle_class3['Curve Length(m)']
plt.hist(x,bins=10,density=True, alpha=0.5, label='Curve
Length(m)',histtype='step',color='grey')
xmin,xmax= plt.xlim()
y=np.linspace(xmin,xmax,100)
p= norm.pdf(y, loc=x.mean(),scale=x.std())
plt.plot(y,p,'k',linewidth=2,label='Normal Distribution')


a,loc,scale=10,0,20
p=gamma.pdf(y,a,loc=loc,scale=scale)
plt.plot(y,p,'r',linewidth=2,label='Gamma Distribution',linestyle='--')

c,loc,scale=10,0,20
p=weibull_min.pdf(y,p,loc=loc, scale=scale)
plt.plot(y,p,'b',linewidth=2,label='Weibull Distribution')

d,loc,scale=10,0,20
p=lognorm.pdf(y,p,loc=loc,scale=scale)
plt.plot(y,p,'g',linewidth=2,label='Lognormal Distribution',linestyle='dashed')
plt.legend()
plt.ylabel('Density')
plt.xlabel('Curve Length(m)')
plt.title('vehicle_class3')
```

Out[37]:

```
Text(0.5, 1.0, 'vehicle class3')
```

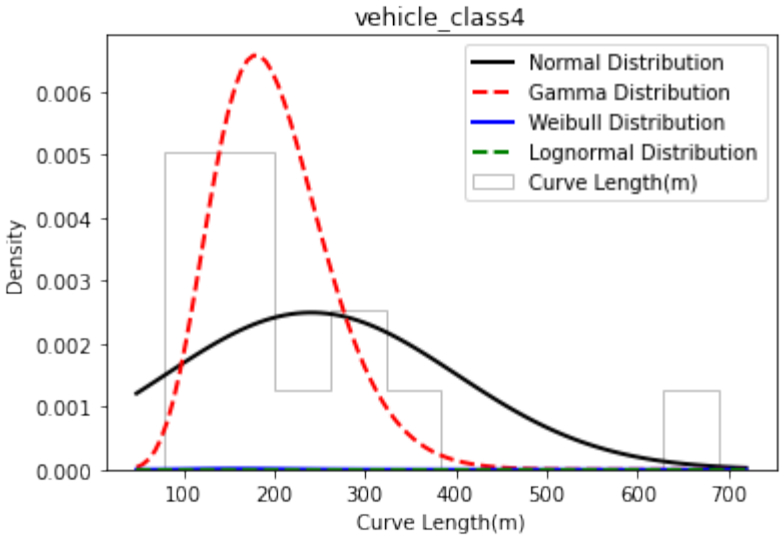


In [38]:

```
x=vehicle_class4['Curve Length(m)']
plt.hist(x,bins=10,density=True, alpha=0.5, label='Curve
Length(m)',histtype='step',color='grey')
xmin,xmax= plt.xlim()
y=np.linspace(xmin,xmax,100)
p= norm.pdf(y, loc=x.mean(),scale=x.std())
plt.plot(y,p,'k',linewidth=2,label='Normal Distribution')


a,loc,scale=10,0,20
p=gamma.pdf(y,a,loc=loc,scale=scale)
plt.plot(y,p,'r',linewidth=2,label='Gamma Distribution',linestyle='--')

c,loc,scale=10,0,20
p=weibull_min.pdf(y,p,loc=loc, scale=scale)
plt.plot(y,p,'b',linewidth=2,label='Weibull Distribution')

d,loc,scale=10,0,20
p=lognorm.pdf(y,p,loc=loc,scale=scale)
plt.plot(y,p,'g',linewidth=2,label='Lognormal Distribution',linestyle='dashed')
plt.legend()
plt.ylabel('Density')
plt.xlabel('Curve Length(m)')
plt.title('vehicle_class4')
```

Out[38]:

```
Text(0.5, 1.0, 'vehicle class4')
```

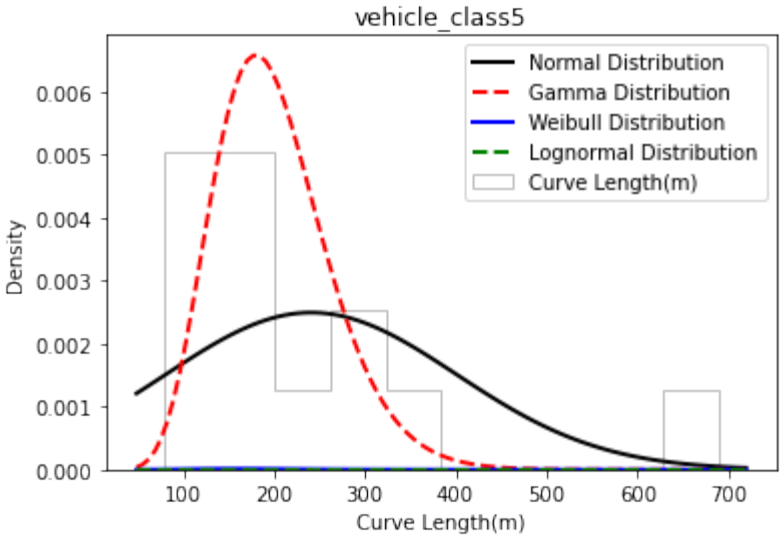


In [39]:

```
x=vehicle_class5['Curve Length(m)']
plt.hist(x,bins=10,density=True, alpha=0.5, label='Curve
Length(m)',histtype='step',color='grey')
xmin,xmax= plt.xlim()
y=np.linspace(xmin,xmax,100)
p= norm.pdf(y, loc=x.mean(),scale=x.std())
plt.plot(y,p,'k',linewidth=2,label='Normal Distribution')


a,loc,scale=10,0,20
p=gamma.pdf(y,a,loc=loc,scale=scale)
plt.plot(y,p,'r',linewidth=2,label='Gamma Distribution',linestyle='--')

c,loc,scale=10,0,20
p=weibull_min.pdf(y,p,loc=loc, scale=scale)
plt.plot(y,p,'b',linewidth=2,label='Weibull Distribution')

d,loc,scale=10,3,200
p=lognorm.pdf(y,p,loc=loc,scale=scale)
plt.plot(y,p,'g',linewidth=2,label='Lognormal Distribution',linestyle='dashed')
plt.legend()
plt.ylabel('Density')
plt.xlabel('Curve Length(m)')
plt.title('vehicle_class5')
```

Out[39]:

```
Text(0.5, 1.0, 'vehicle class5')
```


vehicle_class5
Normal Distribution
Gamma Distribution
Weibull Distribution
Lognormal Distribution
Curve Length(m)
Density
0.006
0.005
0.004
0.003
0.002
0.001
0.000
100
200
300
400
500
600
700
Curve Length(m)

In [ ]: