Prediction of Fuel Emission in Cities using Multiple Linear Regression

Importing Libraries

In [70]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading The Dataset

In [71]:
```
url = r"C:\Users\aswin\Downloads\cars.csv"
df=pd.read_csv(url)
df.head()
```

Out[71]:

s.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Engine Information.Engine Type	Inforr
140	143	202	All-wheel drive	Audi 3.2L 6 cylinder 250hp 236ft-lbs	
140	143	202	Front-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	
140	143	202	Front-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	
140	143	202	All-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	
140	143	202	All-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	

Understanding the dataset

In [72]:
```
df.select_dtypes(include='number').describe()
```

Out[72]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Informa
count	5076.000000	5076.000000	5076.000000	5076.000000	507
mean	145.632191	127.825847	144.012411	5.519110	1
std	62.125026	77.358295	79.925899	0.845637	
min	1.000000	2.000000	1.000000	4.000000	
25%	104.000000	60.000000	62.000000	5.000000	1
50%	152.000000	128.000000	158.000000	6.000000	1
75%	193.000000	198.000000	219.000000	6.000000	2
max	255.000000	255.000000	254.000000	8.000000	3

In [73]:
```
df.select_dtypes(include='object').describe()
```

Out[73]:

	Engine Information.Driveline	Engine Information.Engine Type	Engine Information.Transmission	Fuel Information.Fuel Type	I
count	5076	5076	5076	5076	
unique	4	535	11	4	
top	Rear-wheel drive	Chevrolet 6.2L 8 Cylinder 430 hp 424 ft-lbs	6 Speed Automatic Select Shift	Gasoline	
freq	1751	96	1313	4591	

In [74]:
```
df.shape
```

Out[74]: (5076, 18)

In [75]:
```
df.ndim
```

Out[75]: 2

In [76]:
```
df.dtypes
```

Out[76]:
```
Dimensions.Height                                   int64
Dimensions.Length                                   int64
Dimensions.Width                                    int64
Engine Information.Driveline                       object
Engine Information.Engine Type                     object
Engine Information.Hybrid                            bool
Engine Information.Number of Forward Gears          int64
Engine Information.Transmission                    object
Fuel Information.City mpg                           int64
Fuel Information.Fuel Type                         object
Fuel Information.Highway mpg                        int64
Identification.Classification                      object
Identification.ID                                  object
Identification.Make                                object
Identification.Model Year                          object
Identification.Year                                 int64
Engine Information.Engine Statistics.Horsepower     int64
Engine Information.Engine Statistics.Torque         int64
dtype: object
```

In [77]:
```
categorical_columns=df.select_dtypes(include='object')
numerical_columns=df.select_dtypes(include='number')
```

Data Cleaning

In [78]:
```
#checking for null values
df.isnull().sum()
```

Out[78]:
```
Dimensions.Height                                  0
Dimensions.Length                                  0
Dimensions.Width                                   0
Engine Information.Driveline                       0
Engine Information.Engine Type                     0
Engine Information.Hybrid                          0
Engine Information.Number of Forward Gears         0
Engine Information.Transmission                    0
Fuel Information.City mpg                          0
Fuel Information.Fuel Type                         0
Fuel Information.Highway mpg                       0
Identification.Classification                      0
Identification.ID                                  0
Identification.Make                                0
Identification.Model Year                          0
Identification.Year                                0
Engine Information.Engine Statistics.Horsepower    0
Engine Information.Engine Statistics.Torque        0
dtype: int64
```

In [79]:
```
duplicate_rows=df.duplicated().sum()
print("Number of Duplicate Rows = ",duplicate_rows)
```

```
Number of Duplicate Rows =  18
```

In [80]:
```
#removing duplicate rows
df=df.drop_duplicates()
df
```

Out[80]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informa
0	140	143	202	All-wheel drive	Audi 3 25
1	140	143	202	Front-wheel drive	Audi 2 200
2	140	143	202	Front-wheel drive	Audi 2 200
3	140	143	202	All-wheel drive	Audi 2 200
5	91	17	62	All-wheel drive	Audi 3 26
...	...	...	...	...	
5071	13	253	201	Front-wheel drive	I Cylinde
5072	141	249	108	All-wheel drive	Lambor cylinde
5073	160	249	108	All-wheel drive	Lambor cylinde
5074	200	210	110	Rear-wheel drive	cylind
5075	200	94	110	Rear-wheel drive	cylind

5058 rows × 18 columns

In [81]:
```
df.shape
```

Out[81]:
```
(5058, 18)
```

In [82]:
```
print("Number of Duplicate Rows = ",df.duplicated().sum())
```

```
Number of Duplicate Rows =  0
```

Handling Outliers

In [83]:
```
for col in numerical_columns:
    Q1=df[col].quantile(0.25)
    Q3=df[col].quantile(0.75)
    IQR=Q3-Q1
    lower_bound=Q1-1.5*IQR
    upper_bound=Q3+1.5*IQR
    df=df[(df[col]>=lower_bound) & (df[col]<=upper_bound)]
```

In [84]:
```
df
```

Out[84]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informa
0	140	143	202	All-wheel drive	Audi 3 25
1	140	143	202	Front-wheel drive	Audi 2 200
2	140	143	202	Front-wheel drive	Audi 2 200
3	140	143	202	All-wheel drive	Audi 2 200
5	91	17	62	All-wheel drive	Audi 3 26
...	...	...	...	...	
5069	3	253	201	Front-wheel drive	I Cylinde
5070	3	253	201	Four-wheel drive	I Cylinde
5071	13	253	201	Front-wheel drive	I Cylinde
5074	200	210	110	Rear-wheel drive	cylind
5075	200	94	110	Rear-wheel drive	cylind

4794 rows × 18 columns

In [85]:
```
df=df.drop(columns='Engine Information.Hybrid')
```

In [86]:
```
df
```

Out[86]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informa
0	140	143	202	All-wheel drive	Audi 3 25
1	140	143	202	Front-wheel drive	Audi 2 200
2	140	143	202	Front-wheel drive	Audi 2 200
3	140	143	202	All-wheel drive	Audi 2 200
5	91	17	62	All-wheel drive	Audi 3 265
...	...	...	...	...	
5069	3	253	201	Front-wheel drive	I Cylinde
5070	3	253	201	Four-wheel drive	I Cylinde
5071	13	253	201	Front-wheel drive	I Cylinde
5074	200	210	110	Rear-wheel drive	cylind
5075	200	94	110	Rear-wheel drive	cylind

4794 rows × 17 columns

Encoding

In [87]:
```
categorical_columns.nunique()
```

Out[87]:
```
Engine Information.Driveline          4
Engine Information.Engine Type      535
Engine Information.Transmission      11
Fuel Information.Fuel Type            4
Identification.Classification         2
Identification.ID                  5030
Identification.Make                  47
Identification.Model Year           918
dtype: int64
```

In [88]:
```
target='Fuel Information.City mpg'
catcols_target=['Engine Information.Engine Type','Engine Information.Transm
                'Identification.ID','Identification.Make','Identification.M
catcols_onehot=['Engine Information.Driveline','Fuel Information.Fuel Type'

for i in catcols_target:
    df[i] = df.groupby(i)[target].transform('mean')
```

In [89]:
```
df=pd.get_dummies(df,columns=catcols_onehot)
```

In [90]:
```
df
```

Out[90]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Engine Type	Informati of For
0	140	143	202	18.000000	
1	140	143	202	21.166667	
2	140	143	202	21.166667	
3	140	143	202	21.166667	
5	91	17	62	17.000000	
...	...	...	...	...	
5069	3	253	201	16.800000	
5070	3	253	201	16.800000	
5071	13	253	201	16.800000	
5074	200	210	110	17.000000	
5075	200	94	110	17.000000	

4794 rows × 24 columns

In [91]:
```
df
```

Out[91]:

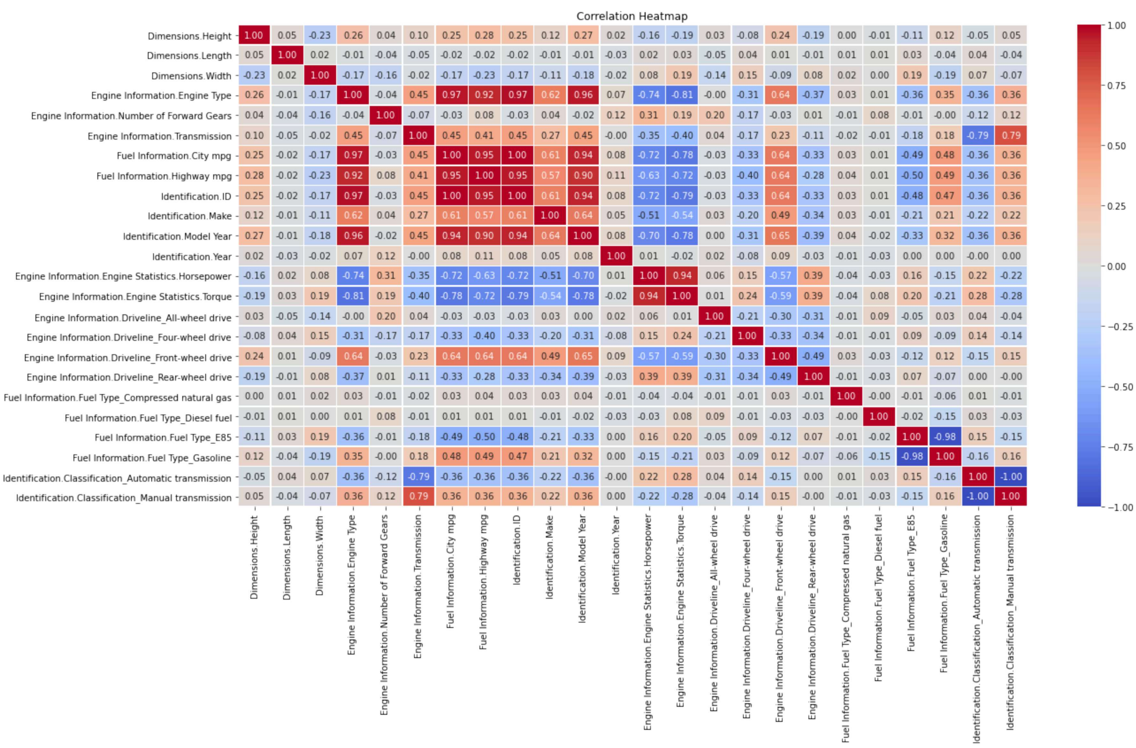
	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Engine Type	Informati of For
0	140	143	202	18.000000	
1	140	143	202	21.166667	
2	140	143	202	21.166667	
3	140	143	202	21.166667	
5	91	17	62	17.000000	
...	...	...	...	...	
5069	3	253	201	16.800000	
5070	3	253	201	16.800000	
5071	13	253	201	16.800000	
5074	200	210	110	17.000000	
5075	200	94	110	17.000000	

4794 rows × 24 columns

In [92]:
```
corr_matrix = df.corr()


plt.figure(figsize=(20,10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths


plt.title('Correlation Heatmap')
plt.show()
```



Scaling

In [93]:
```
scaler=MinMaxScaler()
```

In [94]:
```
df[df.select_dtypes(include='number').columns]=scaler.fit_transform(df[df.s
```

In [95]:
```
df
```

0	0.545455	0.557312	0.794466	0.463415
1	0.545455	0.557312	0.794466	0.617886
2	0.545455	0.557312	0.794466	0.617886
3	0.545455	0.557312	0.794466	0.617886
5	0.351779	0.059289	0.241107	0.414634
...	...	...	...	...
5069	0.003953	0.992095	0.790514	0.404878
5070	0.003953	0.992095	0.790514	0.404878
5071	0.043478	0.992095	0.790514	0.404878
5074	0.782609	0.822134	0.430830	0.414634
5075	0.782609	0.363636	0.430830	0.414634

4794 rows × 24 columns

Feature Selection

In [96]:
```
df.columns
```

Out[96]:
```
Index(['Dimensions.Height', 'Dimensions.Length', 'Dimensions.Width',
       'Engine Information.Engine Type',
       'Engine Information.Number of Forward Gears',
       'Engine Information.Transmission', 'Fuel Information.City mpg',
       'Fuel Information.Highway mpg', 'Identification.ID',
       'Identification.Make', 'Identification.Model Year',
       'Identification.Year',
       'Engine Information.Engine Statistics.Horsepower',
       'Engine Information.Engine Statistics.Torque',
       'Engine Information.Driveline_All-wheel drive',
       'Engine Information.Driveline_Four-wheel drive',
       'Engine Information.Driveline_Front-wheel drive',
       'Engine Information.Driveline_Rear-wheel drive',
       'Fuel Information.Fuel Type_Compressed natural gas',
       'Fuel Information.Fuel Type_Diesel fuel',
       'Fuel Information.Fuel Type_E85', 'Fuel Information.Fuel Type_Gasol
ine',
       'Identification.Classification_Automatic transmission',
       'Identification.Classification_Manual transmission'],
      dtype='object')
```

Selecting features using RFE

In [97]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import Ridge

X = df.drop('Fuel Information.City mpg', axis=1)
y = df['Fuel Information.City mpg']

ridge = Ridge()
rfe = RFE(ridge, n_features_to_select=5)  # Select top 5 features
rfe.fit(X, y)

selected_features = X.columns[rfe.support_]
print("Selected Features by RFE:", selected_features)
```

```
Selected Features by RFE: Index(['Engine Information.Engine Type', 'Fuel I
nformation.Highway mpg',
       'Identification.ID', 'Identification.Model Year',
       'Fuel Information.Fuel Type_E85'],
      dtype='object')
```

VIF (Variance Inflation Factor) and PCA

In [98]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_factor
selected_features = ['Engine Information.Engine Type', 'Fuel Information.Hi
       'Identification.ID',
       'Identification.Model Year',
       'Fuel Information.Fuel Type_E85']
```

In [99]:
```
df[selected_features]
```

Out[99]:

	Engine Information.Engine Type	Fuel Information.Highway mpg	Identification.ID	Identification.Model Year	Informati Ty
0	0.463415	0.482759	0.476190	0.555556	
1	0.617886	0.586207	0.666667	0.555556	
2	0.617886	0.655172	0.619048	0.555556	
3	0.617886	0.586207	0.619048	0.555556	
5	0.414634	0.551724	0.380952	0.346405	
...	...	...	...	...	
5069	0.404878	0.482759	0.476190	0.403922	
5070	0.404878	0.448276	0.428571	0.403922	
5071	0.404878	0.482759	0.476190	0.403922	
5074	0.414634	0.482759	0.428571	0.372549	
5075	0.414634	0.482759	0.428571	0.372549	

4794 rows × 5 columns

In [101]:
```
df['Fuel Information.Fuel Type_E85']=df['Fuel Information.Fuel Type_E85'].a
```

In [102]:
```
df[selected_features]
```

Out[102]:

	Engine Information.Engine Type	Fuel Information.Highway mpg	Identification.ID	Identification.Model Year	Informati Ty
0	0.463415	0.482759	0.476190	0.555556	
1	0.617886	0.586207	0.666667	0.555556	
2	0.617886	0.655172	0.619048	0.555556	
3	0.617886	0.586207	0.619048	0.555556	
5	0.414634	0.551724	0.380952	0.346405	
...	...	...	...	...	
5069	0.404878	0.482759	0.476190	0.403922	
5070	0.404878	0.448276	0.428571	0.403922	
5071	0.404878	0.482759	0.476190	0.403922	
5074	0.414634	0.482759	0.428571	0.372549	
5075	0.414634	0.482759	0.428571	0.372549	

4794 rows × 5 columns

In [105]:
```
# #vif
# from statsmodels.stats.outliers_influence import variance_inflation_facto
# selected_features = ['Engine Information.Engine Type', 'Fuel Information.
#         'Identification.ID',
#         'Identification.Model Year',
#         'Fuel Information.Fuel Type_E85']

X = df[selected_features]
#Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.

print(vif_data)
```

```
                          Feature         VIF
0  Engine Information.Engine Type  153.036549
1    Fuel Information.Highway mpg   62.119350
2               Identification.ID  179.245052
3       Identification.Model Year   51.423572
4  Fuel Information.Fuel Type_E85    1.282206
```

In [106]:
```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features = ['Engine Information.Engine Type', 'Fuel Information.Hi


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features])

# Create a new column in the DataFrame for the PCA component
df['PCA_Enginetype_highway'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features)

print("Transformed DataFrame with PCA feature added:")
df.head()
```

```
Transformed DataFrame with PCA feature added:
```

Out[106]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Information
0	0.545455	0.557312	0.794466	0.666667	
1	0.545455	0.557312	0.794466	0.666667	
2	0.545455	0.557312	0.794466	0.666667	
3	0.545455	0.557312	0.794466	0.666667	
5	0.351779	0.059289	0.241107	0.666667	

5 rows × 23 columns

In [107]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_factor
selected_features = ['PCA_Enginetype_highway',
       'Identification.ID',
       'Identification.Model Year',
       'Fuel Information.Fuel Type_E85']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.

print(vif_data)
```

```
                          Feature        VIF
0          PCA_Enginetype_highway   2.122976
1               Identification.ID  45.165115
2       Identification.Model Year  50.004440
3  Fuel Information.Fuel Type_E85   1.673974
```

In [108]:
```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features1 = ['Identification.ID', 'Identification.Model Year']


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features1])

# Create a new column in the DataFrame for the PCA component
df['PCA_ID_Model Year'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features1)

print("Transformed DataFrame with PCA feature added:")
df.head()
```

```
Transformed DataFrame with PCA feature added:
```

Out[108]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Information
0	0.545455	0.557312	0.794466	0.666667	
1	0.545455	0.557312	0.794466	0.666667	
2	0.545455	0.557312	0.794466	0.666667	
3	0.545455	0.557312	0.794466	0.666667	
5	0.351779	0.059289	0.241107	0.666667	

5 rows × 22 columns

In [109]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_factor
selected_features = ['PCA_Enginetype_highway',
       'PCA_ID_Model Year',
       'Fuel Information.Fuel Type_E85']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.

print(vif_data)
```

```
                          Feature        VIF
0          PCA_Enginetype_highway  23.623078
1               PCA_ID_Model Year  23.140748
2  Fuel Information.Fuel Type_E85   1.208917
```

In [110]:
```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features2 = ['PCA_Enginetype_highway', 'PCA_ID_Model Year']


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features2])

# Create a new column in the DataFrame for the PCA component
df['PCA_Final'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features2)

print("Transformed DataFrame with PCA feature added:")
df.head()
```

```
Transformed DataFrame with PCA feature added:
```

Out[110]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Information
0	0.545455	0.557312	0.794466	0.666667	
1	0.545455	0.557312	0.794466	0.666667	
2	0.545455	0.557312	0.794466	0.666667	
3	0.545455	0.557312	0.794466	0.666667	
5	0.351779	0.059289	0.241107	0.666667	

5 rows × 21 columns

In [111]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_factor
selected_features = [
       'PCA_Final',
       'Fuel Information.Fuel Type_E85']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.

print(vif_data)
```

```
                          Feature       VIF
0                       PCA_Final  1.192972
1  Fuel Information.Fuel Type_E85  1.192972
```

Hence, the features are multicollinear

Model Training

In [112]:
```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

model=LinearRegression()

X=df[['PCA_Final','Fuel Information.Fuel Type_E85']]
Y=df[['Fuel Information.City mpg']]

model.fit(X,Y)
pred=model.predict(X)
```

Calculating R^2 value

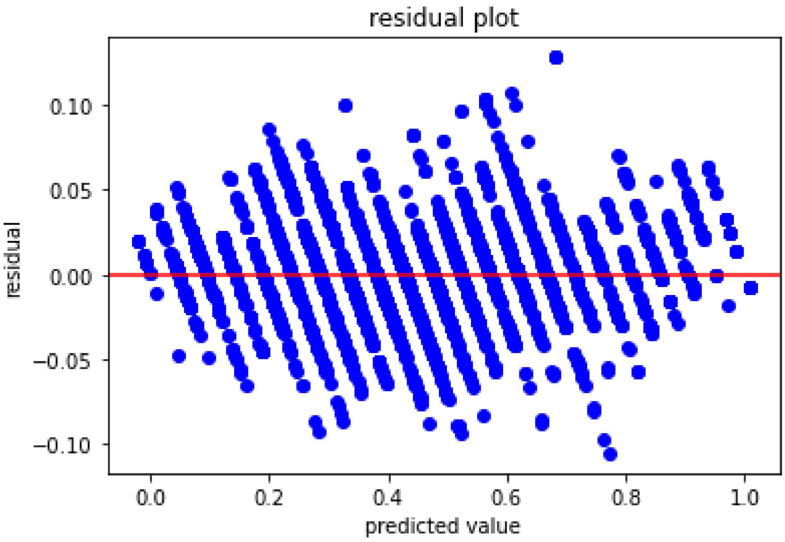
In [113]:
```
from sklearn.metrics import r2_score
r2_score_value = r2_score(Y, pred)
print(r2_score_value)
```

```
0.9814751698519412
```

Homoscedasticity

In [114]:
```
err=Y-pred
plt.scatter(pred,err,color='b')
plt.xlabel('predicted value')
plt.ylabel('residual')
plt.title('residual plot')
plt.axhline(y=0,color='r')
```

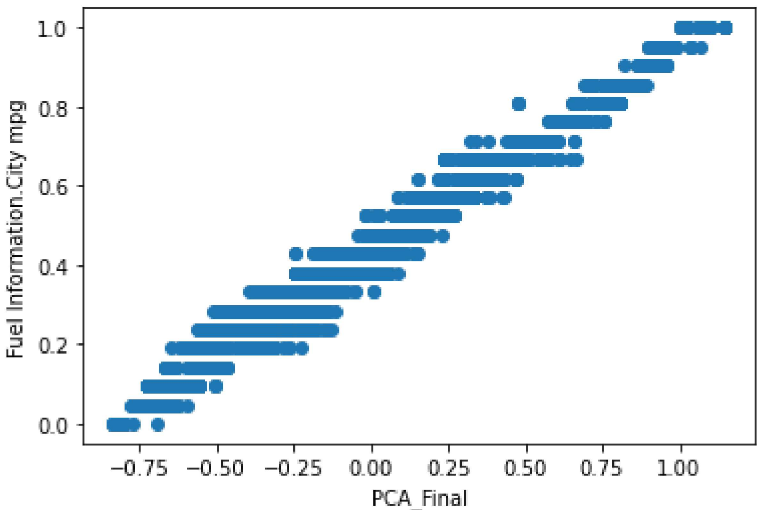
Out[114]: <matplotlib.lines.Line2D at 0x1cf6242b4f0>



Linearity

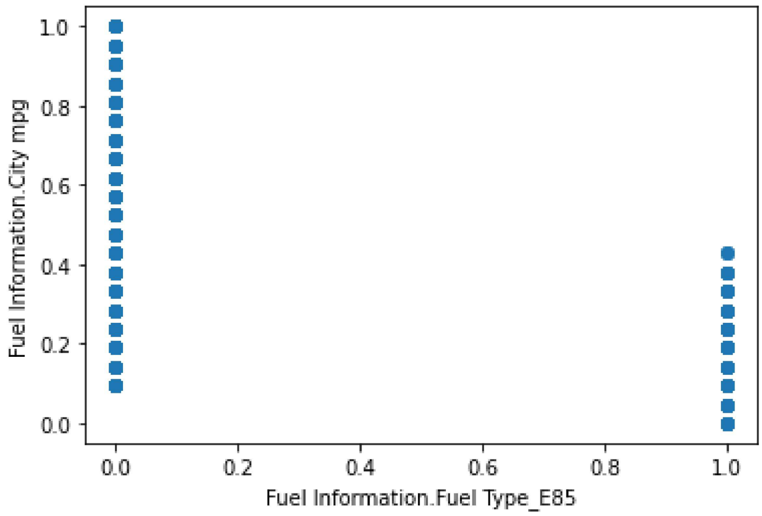
In [115]:
```
x=df['PCA_Final']
y=df['Fuel Information.City mpg']
plt.xlabel('PCA_Final')
plt.ylabel('Fuel Information.City mpg')
plt.scatter(x,y)
```

Out[115]: <matplotlib.collections.PathCollection at 0x1cf630b7850>



In [116]:
```
x=df['Fuel Information.Fuel Type_E85']
y=df['Fuel Information.City mpg']
plt.xlabel('Fuel Information.Fuel Type_E85')
plt.ylabel('Fuel Information.City mpg')
plt.scatter(x,y)
```

Out[116]: <matplotlib.collections.PathCollection at 0x1cf6250f5b0>



Conclusion

The cars dataset was analysed, preprocessed, categorical variables encoded using various encoding techniques. Required features were selected using RFE and Dimentionality reduction was done using PCA to reduce the VIF value.

Model Training was done using Linear Regression model and Residual plot was plotted

Graphs were plotted to verify the assumptions like linearity and homoscedasticity