# Model Building

| | |
|---|---|
| Team ID | PNT2022TMID12635 |
| Project Name | Machine Learning based Vehicle Performance Analyzer |

# Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

# Importing Dataset

In [2]:

```
dataset=pd.read_csv('/content/car performance.csv')
dataset
```

Out[2]:

| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin | car name |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130 | 3504 | 12.0 | 70 | 1 | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165 | 3693 | 11.5 | 70 | 1 | buick skylark 320 |
| **2** | 18.0 | 8 | 318.0 | 150 | 3436 | 11.0 | 70 | 1 | plymouth satellite |
| **3** | 16.0 | 8 | 304.0 | 150 | 3433 | 12.0 | 70 | 1 | amc rebel sst |
| **4** | 17.0 | 8 | 302.0 | 140 | 3449 | 10.5 | 70 | 1 | ford torino |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **393** | 27.0 | 4 | 140.0 | 86 | 2790 | 15.6 | 82 | 1 | ford mustang gl |
| **394** | 44.0 | 4 | 97.0 | 52 | 2130 | 24.6 | 82 | 2 | vw pickup |
| **395** | 32.0 | 4 | 135.0 | 84 | 2295 | 11.6 | 82 | 1 | dodge rampage |
| **396** | 28.0 | 4 | 120.0 | 79 | 2625 | 18.6 | 82 | 1 | ford ranger |
| **397** | 31.0 | 4 | 119.0 | 82 | 2720 | 19.4 | 82 | 1 | chevy s-10 |

398 rows × 9 columns

# Finding missing data

In [3]:

```
dataset.isnull().any()
```

Out[3]:

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model year      False
origin          False
car name        False
dtype: bool
```

**There are no null characters in the columns but there is a special character '?' in the 'horsepower' column. So we we replaced '?' with nan and replaced nan values with mean of the column.**

In [4]:

```
dataset['horsepower']=dataset['horsepower'].replace('?',np.nan)
```

In [5]:

```
dataset['horsepower'].isnull().sum()
```

Out[5]:

```
6
```

In [6]:

```
dataset['horsepower']=dataset['horsepower'].astype('float64')
```

In [7]:

```
dataset['horsepower'].fillna((dataset['horsepower'].mean()),inplace=True)
```

In [8]:

```
dataset.isnull().any()
```

Out[8]:

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model year      False
origin          False
car name        False
dtype: bool
```

In [9]:

```
dataset.info() #Pandas dataframe.info() function is used to get a quick overview of the d
ataset.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

In [10]:

```
dataset.describe() #Pandas describe() is used to view some basic statistical details of a
data frame or a series of numeric values.
```

Out[10]:

| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin |
|---|---|---|---|---|---|---|---|---|
| **count** | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 |
| **mean** | 23.514573 | 5.454774 | 193.425879 | 104.469388 | 2970.424623 | 15.568090 | 76.010050 | 1.572864 |

| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin |
|---|---|---|---|---|---|---|---|---|
| std | 7.815984 | 1.701004 | 104.269838 | 38.199187 | 846.841774 | 2.757689 | 3.697627 | 0.802055 |
| min | 9.000000 | 3.000000 | 68.000000 | 46.000000 | 1613.000000 | 8.000000 | 70.000000 | 1.000000 |
| 25% | 17.500000 | 4.000000 | 104.250000 | 76.000000 | 2223.750000 | 13.825000 | 73.000000 | 1.000000 |
| 50% | 23.000000 | 4.000000 | 148.500000 | 95.000000 | 2803.500000 | 15.500000 | 76.000000 | 1.000000 |
| 75% | 29.000000 | 8.000000 | 262.000000 | 125.000000 | 3608.000000 | 17.175000 | 79.000000 | 2.000000 |
| max | 46.600000 | 8.000000 | 455.000000 | 230.000000 | 5140.000000 | 24.800000 | 82.000000 | 3.000000 |

**There is no use with car name attribute so drop it**

In [11]:

```
dataset=dataset.drop('car name',axis=1) #dropping the unwanted column.
```

In [12]:

```
corr_table=dataset.corr()#Pandas dataframe.corr() is used to find the pairwise correlatio
n of all columns in the dataframe.
corr_table
```

Out[12]:

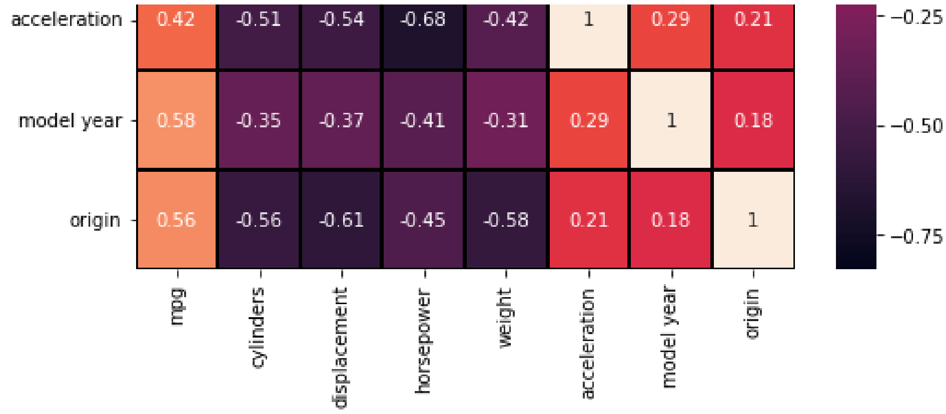
| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin |
|---|---|---|---|---|---|---|---|---|
| mpg | 1.000000 | -0.775396 | -0.804203 | -0.771437 | -0.831741 | 0.420289 | 0.579267 | 0.563450 |
| cylinders | -0.775396 | 1.000000 | 0.950721 | 0.838939 | 0.896017 | -0.505419 | -0.348746 | -0.562543 |
| displacement | -0.804203 | 0.950721 | 1.000000 | 0.893646 | 0.932824 | -0.543684 | -0.370164 | -0.609409 |
| horsepower | -0.771437 | 0.838939 | 0.893646 | 1.000000 | 0.860574 | -0.684259 | -0.411651 | -0.453669 |
| weight | -0.831741 | 0.896017 | 0.932824 | 0.860574 | 1.000000 | -0.417457 | -0.306564 | -0.581024 |
| acceleration | 0.420289 | -0.505419 | -0.543684 | -0.684259 | -0.417457 | 1.000000 | 0.288137 | 0.205873 |
| model year | 0.579267 | -0.348746 | -0.370164 | -0.411651 | -0.306564 | 0.288137 | 1.000000 | 0.180662 |
| origin | 0.563450 | -0.562543 | -0.609409 | -0.453669 | -0.581024 | 0.205873 | 0.180662 | 1.000000 |

# Data Visualizations

**Heatmap : which represents correlation between attributes**

In [13]:

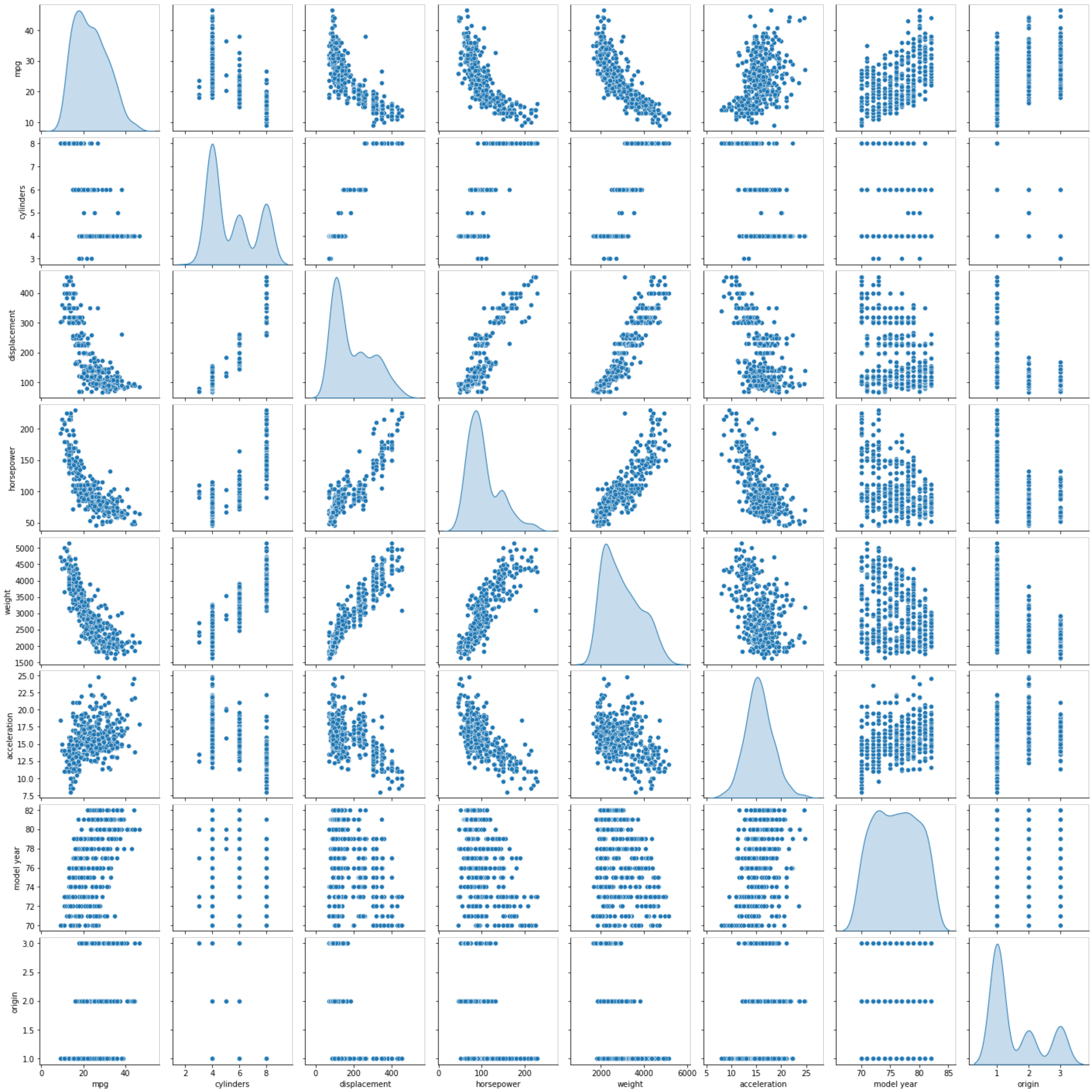
```
sns.heatmap(dataset.corr(),annot=True,linecolor ='black', linewidths = 1)#Heatmap is a wa
y to show some sort of matrix plot,annot is used for correlation.
fig=plt.gcf()
fig.set_size_inches(8,8)
```




**Visualizations of each attributes w.r.t rest of all attributes**

In [14]:

```
sns.pairplot(dataset,diag_kind='kde') #pairplot represents pairwise relation across the e
ntire dataframe.
plt.show()
```




**Regression plots(regplot()) creates a regression line between 2 parameters and helps to visualize their linear**

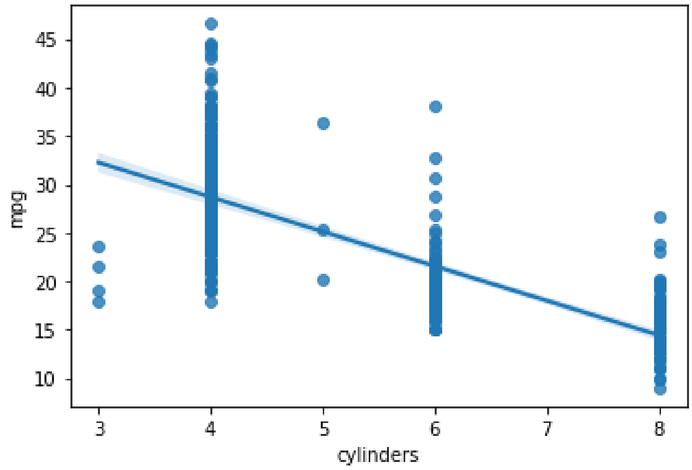
**relationships.**

In [15]:

```
sns.regplot(x="cylinders", y="mpg", data=dataset)
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092aab8750>

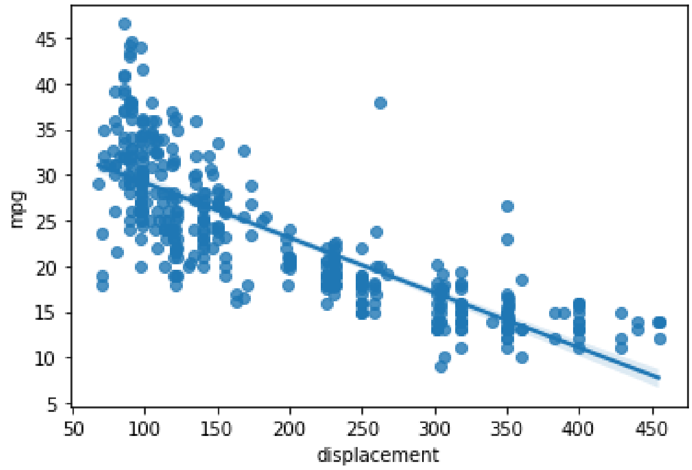



In [16]:

```
sns.regplot(x="displacement", y="mpg", data=dataset)
```

Out[16]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092920a750>

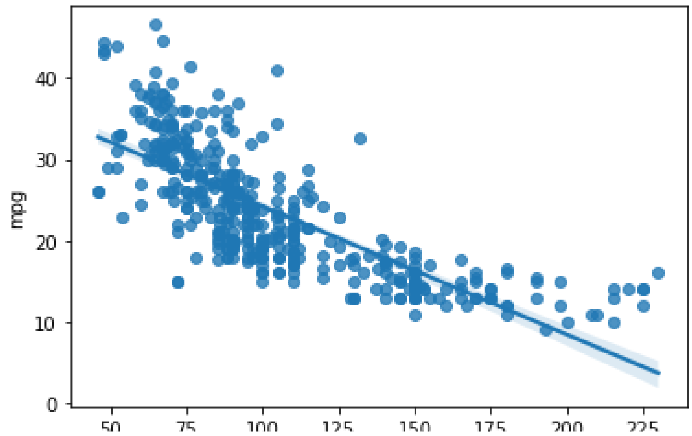



In [17]:

```
sns.regplot(x="horsepower", y="mpg", data=dataset)
```

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092ab6c790>

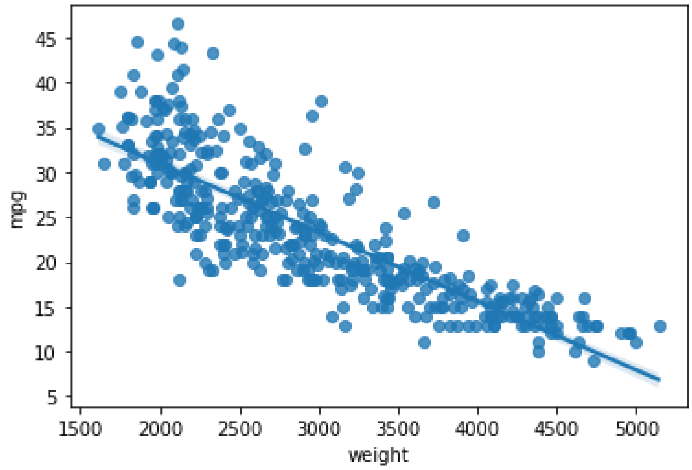



In [18]:

```
sns.regplot(x="weight", y="mpg", data=dataset)
```

Out[18]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092aa32710>

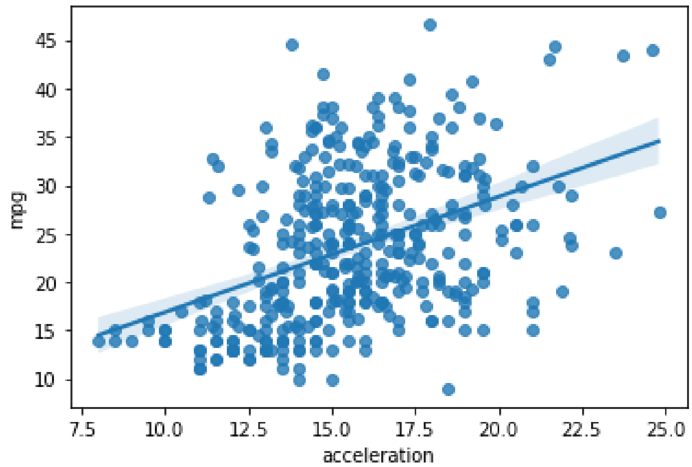



In [19]:

```
sns.regplot(x="acceleration", y="mpg", data=dataset)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092abd9990>

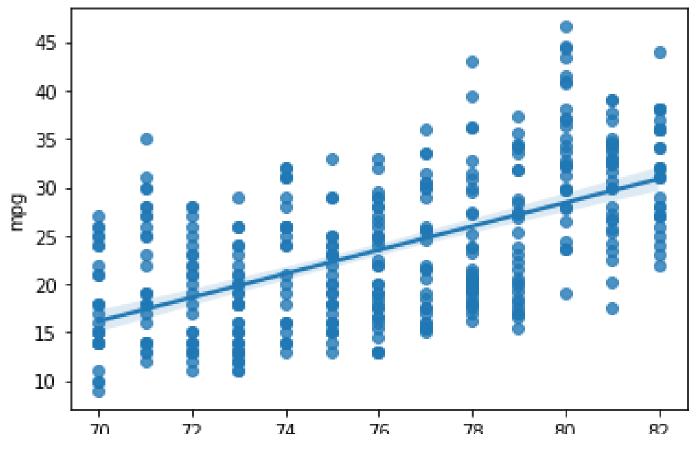


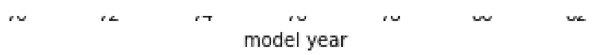

In [20]:

```
sns.regplot(x="model year", y="mpg", data=dataset)
```

Out[20]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092aba7410>



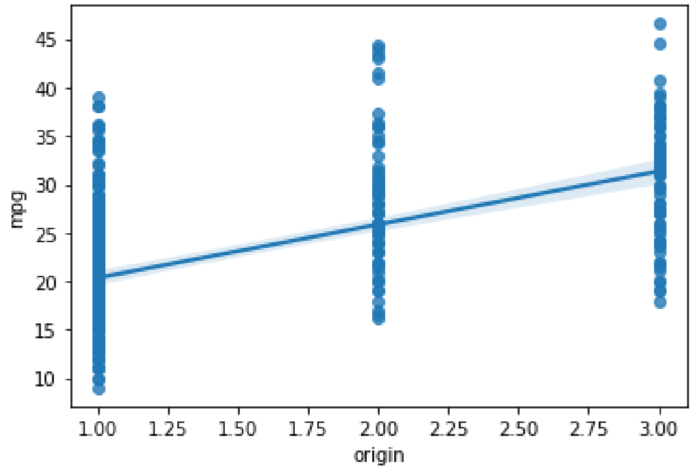



In [21]:

```
sns.regplot(x="origin", y="mpg", data=dataset)
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092ab3c450>

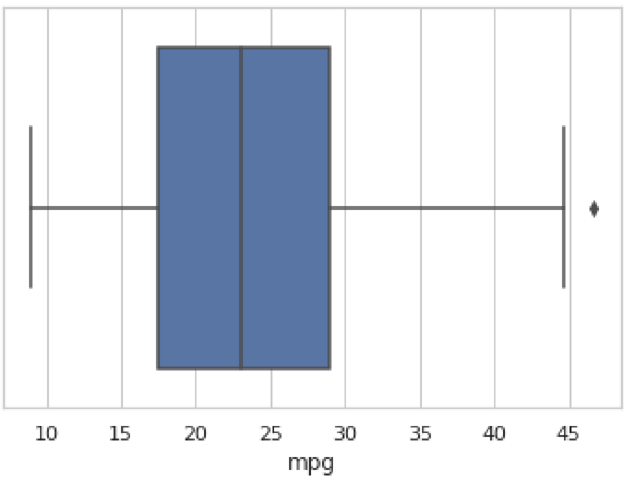



In [22]:

```
sns.set(style="whitegrid")
sns.boxplot(x=dataset["mpg"])
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f092ab63f10>




**Finding quartiles for mgp**

# The P-value is the probability value that the correlation between these two variables is statistically significant.

**Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.**

**By convention, when the**

- **p-value is $<$ 0.001: we say there is strong evidence that the correlation is significant.**
- **the p-value is $<$ 0.05: there is moderate evidence that the correlation is significant.**
- **the p-value is $<$ 0.1: there is weak evidence that the correlation is significant.**
- **the p-value is $>$ 0.1: there is no evidence that the correlation is significant.**

In [23]:

In [23]:

```
from scipy import stats
```

## Cylinders vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'Cylinders' and 'mpg'.**

In [24]:

```
pearson_coef, p_value = stats.pearsonr(dataset['cylinders'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is -0.7753962854205542  with a P-value of P = 4.50399
2246177055e-81
```

***Conclusion:***

**Since the p-value is $<$ 0.001, the correlation between cylinders and mpg is statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.**

## Displacement vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'Displacement' and 'mpg'.**

In [25]:

```
pearson_coef, p_value = stats.pearsonr(dataset['displacement'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is -0.8042028248058978  with a P-value of P = 1.65588
89101930157e-91
```

***Conclusion:***

**Since the p-value is $<$ 0.1, the correlation between displacement and mpg is statistically significant, and the linear negative relationship is quite strong (~-0.809, close to -1)**

## Horsepower vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'horsepower' and 'mpg'.**

In [26]:

```
pearson_coef, p_value = stats.pearsonr(dataset['horsepower'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is -0.7714371350025526  with a P-value of P = 9.25547
7533166725e-80
```

***Conclusion:***

**Since the p-value is $<$ 0.001, the correlation between horsepower and mpg is statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative and moderately strong.**

## Weight vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'weight' and 'mpg'.**

In [28]:

```
pearson_coef, p_value = stats.pearsonr(dataset['weight'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is -0.831740933244335  with a P-value of P = 2.972799
5640500577e-103
```

***Conclusion:***

**Since the p-value is $<$ 0.001, the correlation between weight and mpg is statistically significant, and the linear negative relationship is quite strong (~-0.831, close to -1)**

### Acceleration vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'Acceleration' and 'mpg'.**

In [29]:

```
pearson_coef, p_value = stats.pearsonr(dataset['acceleration'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is 0.4202889121016507  with a P-value of P = 1.823091
535078553e-18
```

***Conclusion:***

**Since the p-value is $>$ 0.1, the correlation between acceleration and mpg is statistically significant, but the linear relationship is weak (~0.420).**

### Model year vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'Model year' and 'mpg'.**

In [30]:

```
pearson_coef, p_value = stats.pearsonr(dataset['model year'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is 0.5792671330833096  with a P-value of P = 4.844935
813365483e-37
```

***Conclusion:***

**Since the p-value is $<$ 0.001, the correlation between model year and mpg is statistically significant, but the linear relationship is only moderate (~0.579).**

### Origin vs mpg

**Let's calculate the Pearson Correlation Coefficient and P-value of 'Origin' and 'mpg'.**

In [31]:

```
pearson_coef, p_value = stats.pearsonr(dataset['origin'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p
_value)
```

```
The Pearson Correlation Coefficient is 0.5634503597738431  with a P-value of P = 1.011482
2102336483e-34
```

***Conclusion:***

**Since the p-value is $< 0.001$, the correlation between origin and mpg is statistically significant, but the linear relationship is only moderate (~0.563).**

**Ordinary Least Squares Statistics**

In [32]:

```
test=smf.ols('mpg~cylinders+displacement+horsepower+weight+acceleration+origin',dataset).
fit()
test.summary()
```

Out[32]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | mpg | **R-squared:** | 0.717 |
| **Model:** | OLS | **Adj. R-squared:** | 0.713 |
| **Method:** | Least Squares | **F-statistic:** | 165.5 |
| **Date:** | Mon, 07 Nov 2022 | **Prob (F-statistic):** | 4.84e-104 |
| **Time:** | 09:56:04 | **Log-Likelihood:** | -1131.1 |
| **No. Observations:** | 398 | **AIC:** | 2276. |
| **Df Residuals:** | 391 | **BIC:** | 2304. |
| **Df Model:** | 6 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 42.7111 | 2.693 | 15.861 | 0.000 | 37.417 | 48.005 |
| **cylinders** | -0.5256 | 0.404 | -1.302 | 0.194 | -1.320 | 0.268 |
| **displacement** | 0.0106 | 0.009 | 1.133 | 0.258 | -0.008 | 0.029 |
| **horsepower** | -0.0529 | 0.016 | -3.277 | 0.001 | -0.085 | -0.021 |
| **weight** | -0.0051 | 0.001 | -6.441 | 0.000 | -0.007 | -0.004 |
| **acceleration** | 0.0043 | 0.120 | 0.036 | 0.972 | -0.232 | 0.241 |
| **origin** | 1.4269 | 0.345 | 4.136 | 0.000 | 0.749 | 2.105 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 32.659 | **Durbin-Watson:** | 0.886 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 43.338 |
| **Skew:** | 0.624 | **Prob(JB):** | 3.88e-10 |
| **Kurtosis:** | 4.028 | **Cond. No.** | 3.99e+04 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.99e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

**Inference as in the above summary the p value of the accelaration is maximum(i.e 0.972) so we can remove the acc variable from the dataset**

# Seperating into Dependent and Independent variables

**Independent variables**

In [33]:

```
x=dataset[['cylinders','displacement','horsepower','weight','model year','origin']].valu
es
x
```

Out[33]:

```
array([[8.000e+00, 3.070e+02, 1.300e+02, 3.504e+03, 7.000e+01, 1.000e+00],
       [8.000e+00, 3.500e+02, 1.650e+02, 3.693e+03, 7.000e+01, 1.000e+00],
       [8.000e+00, 3.180e+02, 1.500e+02, 3.436e+03, 7.000e+01, 1.000e+00],
       ...,
       [4.000e+00, 1.350e+02, 8.400e+01, 2.295e+03, 8.200e+01, 1.000e+00],
       [4.000e+00, 1.200e+02, 7.900e+01, 2.625e+03, 8.200e+01, 1.000e+00],
       [4.000e+00, 1.190e+02, 8.200e+01, 2.720e+03, 8.200e+01, 1.000e+00]])
```

### Dependent variables

In [34]:

```
y=dataset.iloc[:,0:1].values
y
```

Out[34]:

```
array([[18. ],
       [15. ],
       [18. ],
       [16. ],
       [17. ],
       [15. ],
       [14. ],
       [14. ],
       [14. ],
       [15. ],
       [15. ],
       [14. ],
       [15. ],
       [14. ],
       [24. ],
       [22. ],
       [18. ],
       [21. ],
       [27. ],
       [26. ],
       [25. ],
       [24. ],
       [25. ],
       [26. ],
       [21. ],
       [10. ],
       [10. ],
       [11. ],
       [ 9. ],
       [27. ],
       [28. ],
       [25. ],
       [25. ],
       [19. ],
       [16. ],
       [17. ],
       [19. ],
       [18. ],
       [14. ],
       [14. ],
       [14. ],
       [14. ],
       [12. ],
       [13. ],
       [13. ],
       [18. ],
       [22. ],
```

```
       [22. ],
       [19. ],
       [18. ],
       [23. ],
       [28. ],
       [30. ],
       [30. ],
       [31. ],
       [35. ],
       [27. ],
       [26. ],
       [24. ],
       [25. ],
       [23. ],
       [20. ],
       [21. ],
       [13. ],
       [14. ],
       [15. ],
       [14. ],
       [17. ],
       [11. ],
       [13. ],
       [12. ],
       [13. ],
       [19. ],
       [15. ],
       [13. ],
       [13. ],
       [14. ],
       [18. ],
       [22. ],
       [21. ],
       [26. ],
       [22. ],
       [28. ],
       [23. ],
       [28. ],
       [27. ],
       [13. ],
       [14. ],
       [13. ],
       [14. ],
       [15. ],
       [12. ],
       [13. ],
       [13. ],
       [14. ],
       [13. ],
       [12. ],
       [13. ],
       [18. ],
       [16. ],
       [18. ],
       [18. ],
       [23. ],
       [26. ],
       [11. ],
       [12. ],
       [13. ],
       [12. ],
       [18. ],
       [20. ],
       [21. ],
       [22. ],
       [18. ],
       [19. ],
       [21. ],
       [26. ],
       [15. ],
       [16. ],
       [29. ],
       [24. ],
```

```
       [21. ],
       [20. ],
       [19. ],
       [15. ],
       [24. ],
       [20. ],
       [11. ],
       [20. ],
       [21. ],
       [19. ],
       [15. ],
       [31. ],
       [26. ],
       [32. ],
       [25. ],
       [16. ],
       [16. ],
       [18. ],
       [16. ],
       [13. ],
       [14. ],
       [14. ],
       [14. ],
       [29. ],
       [26. ],
       [26. ],
       [31. ],
       [32. ],
       [28. ],
       [24. ],
       [26. ],
       [24. ],
       [26. ],
       [31. ],
       [19. ],
       [18. ],
       [15. ],
       [15. ],
       [16. ],
       [15. ],
       [16. ],
       [14. ],
       [17. ],
       [16. ],
       [15. ],
       [18. ],
       [21. ],
       [20. ],
       [13. ],
       [29. ],
       [23. ],
       [20. ],
       [23. ],
       [24. ],
       [25. ],
       [24. ],
       [18. ],
       [29. ],
       [19. ],
       [23. ],
       [23. ],
       [22. ],
       [25. ],
       [33. ],
       [28. ],
       [25. ],
       [25. ],
       [26. ],
       [27. ],
       [17.5],
       [16. ],
       [15.5],
       [14.5],
```

```
       [22. ],
       [22. ],
       [24. ],
       [22.5],
       [29. ],
       [24.5],
       [29. ],
       [33. ],
       [20. ],
       [18. ],
       [18.5],
       [17.5],
       [29.5],
       [32. ],
       [28. ],
       [26.5],
       [20. ],
       [13. ],
       [19. ],
       [19. ],
       [16.5],
       [16.5],
       [13. ],
       [13. ],
       [13. ],
       [31.5],
       [30. ],
       [36. ],
       [25.5],
       [33.5],
       [17.5],
       [17. ],
       [15.5],
       [15. ],
       [17.5],
       [20.5],
       [19. ],
       [18.5],
       [16. ],
       [15.5],
       [15.5],
       [16. ],
       [29. ],
       [24.5],
       [26. ],
       [25.5],
       [30.5],
       [33.5],
       [30. ],
       [30.5],
       [22. ],
       [21.5],
       [21.5],
       [43.1],
       [36.1],
       [32.8],
       [39.4],
       [36.1],
       [19.9],
       [19.4],
       [20.2],
       [19.2],
       [20.5],
       [20.2],
       [25.1],
       [20.5],
       [19.4],
       [20.6],
       [20.8],
       [18.6],
       [18.1],
       [19.2],
```

```
[19.2],
[17.7],
[18.1],
[17.5],
[30. ],
[27.5],
[27.2],
[30.9],
[21.1],
[23.2],
[23.8],
[23.9],
[20.3],
[17. ],
[21.6],
[16.2],
[31.5],
[29.5],
[21.5],
[19.8],
[22.3],
[20.2],
[20.6],
[17. ],
[17.6],
[16.5],
[18.2],
[16.9],
[15.5],
[19.2],
[18.5],
[31.9],
[34.1],
[35.7],
[27.4],
[25.4],
[23. ],
[27.2],
[23.9],
[34.2],
[34.5],
[31.8],
[37.3],
[28.4],
[28.8],
[26.8],
[33.5],
[41.5],
[38.1],
[32.1],
[37.2],
[28. ],
[26.4],
[24.3],
[19.1],
[34.3],
[29.8],
[31.3],
[37. ],
[32.2],
[46.6],
[27.9],
[40.8],
[44.3],
[43.4],
[36.4],
[30. ],
[44.6],
[40.9],
[33.8],
[29.8],
[32.7],
[23.7],
```

```
       [35. ],
       [23.6],
       [32.4],
       [27.2],
       [26.6],
       [25.8],
       [23.5],
       [30. ],
       [39.1],
       [39. ],
       [35.1],
       [32.3],
       [37. ],
       [37.7],
       [34.1],
       [34.7],
       [34.4],
       [29.9],
       [33. ],
       [34.5],
       [33.7],
       [32.4],
       [32.9],
       [31.6],
       [28.1],
       [30.7],
       [25.4],
       [24.2],
       [22.4],
       [26.6],
       [20.2],
       [17.6],
       [28. ],
       [27. ],
       [34. ],
       [31. ],
       [29. ],
       [27. ],
       [24. ],
       [23. ],
       [36. ],
       [37. ],
       [31. ],
       [38. ],
       [36. ],
       [36. ],
       [36. ],
       [34. ],
       [38. ],
       [32. ],
       [38. ],
       [25. ],
       [38. ],
       [26. ],
       [22. ],
       [32. ],
       [36. ],
       [27. ],
       [27. ],
       [44. ],
       [32. ],
       [28. ],
       [31. ]])
```

## Splitting into train and test data.

In [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
```

**we are splitting as 90% train data and 10% test data**

# Normalisation

In [ ]:

```
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()
x_train = sd.fit_transform(x_train)
x_test = sd.fit_transform(x_test)
y_train = sd.fit_transform(y_train)
y_test = sd.fit_transform(y_test)

x_train
```

# decision tree regressor

In [37]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(random_state=0,criterion="mae")
dt.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py:370: FutureWarning: Crite
rion 'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='abs
olute_error'` which is equivalent.
  FutureWarning,
```

Out[37]:

```
DecisionTreeRegressor(criterion='mae', random_state=0)
```

In [38]:

```
import pickle
pickle.dump(dt,open('decision_model.pkl','wb'))
```

In [39]:

```
y_pred=dt.predict(x_test)
y_pred
```

Out[39]:

```
array([14. , 26.5, 14. , 19. , 18. , 31. , 37. , 22. , 15. , 26.8, 34.5,
       31.8, 16. , 26. , 15.5, 34.3, 26.6, 27. , 16. , 31.5, 16. , 23. ,
       27.5, 19. , 33.8, 24.2, 36.1, 36. , 34.3, 18.5, 19.2, 35. , 15. ,
       32. , 22. , 23. , 19.4, 16. , 31.5, 12. ])
```

In [40]:

```
y_test
```

Out[40]:

```
array([[14. ],
       [25. ],
       [13. ],
       [21. ],
       [18. ],
       [35. ],
       [34.1],
       [20. ],
       [15. ],
```

```
       [23.5],
       [40.9],
       [37.2],
       [18. ],
       [23. ],
       [15.5],
       [35.7],
       [31. ],
       [27. ],
       [18. ],
       [37.3],
       [15.5],
       [23. ],
       [24. ],
       [18. ],
       [34.5],
       [25.4],
       [36.1],
       [34. ],
       [30. ],
       [16. ],
       [18.6],
       [37. ],
       [15. ],
       [33.5],
       [22.4],
       [24. ],
       [19. ],
       [16.9],
       [31.9],
       [12. ]])
```

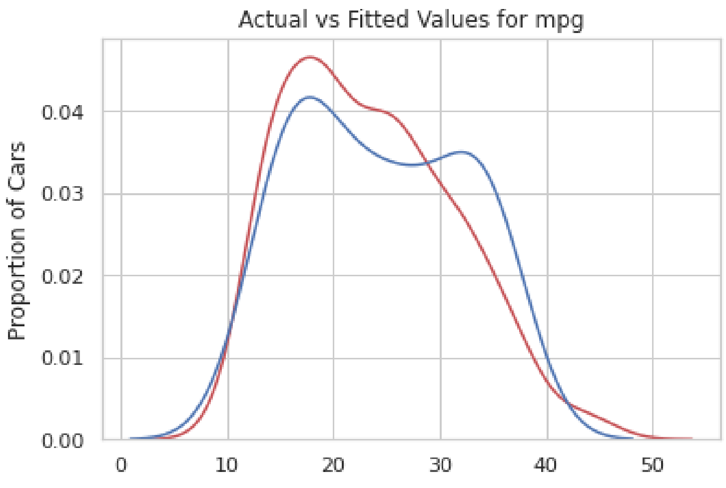
In [60]:

```
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred, hist=False, color="b", label="Fitted Values" , ax=ax1)


plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `kde
plot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `kde
plot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.

**R-squared**

**R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.**

**R-squared = Explained variation / Total variation**

**Mean Squared Error (MSE)**

**The Mean Squared Error measures the average of the squares of errors, that is, the difference between actual value (y) and the estimated value (ŷ).**

In [42]:

```
from sklearn.metrics import r2_score,mean_squared_error
```

In [43]:

```
r2_score(y_test,y_pred)
```

Out[43]:

```
0.912578781275149
```

In [44]:

```
mean_squared_error(y_test,y_pred)
```

Out[44]:

```
6.042499999999999
```

In [45]:

```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[45]:

```
2.458149710656371
```

# random forest regressor

In [46]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [47]:

```
rf= RandomForestRegressor(n_estimators=10,random_state=0,criterion='mae')
rf.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().

/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py:407: FutureWarning: Cr
iterion 'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='
absolute_error'` which is equivalent.
  FutureWarning,
```

Out[47]:

```
RandomForestRegressor(criterion='mae', n_estimators=10, random_state=0)
```

In [48]:

```
y_pred2=rf.predict(x_test)
y_pred2
```

Out[48]:

```
array([14.05, 25.55, 13.7 , 21.2 , 18.5 , 30.8 , 34.31, 22.5 , 15.1 ,
       24.46, 32.01, 39.79, 17.8 , 24.85, 15.85, 31.31, 28.32, 26.93,
       16.54, 32.12, 16.05, 25.6 , 23.86, 20.56, 32.19, 24.75, 32.39,
       32.64, 31.02, 16.39, 18.35, 30.1 , 17.67, 31.  , 22.91, 23.5 ,
       19.77, 16.1 , 33.65, 12.  ])
```

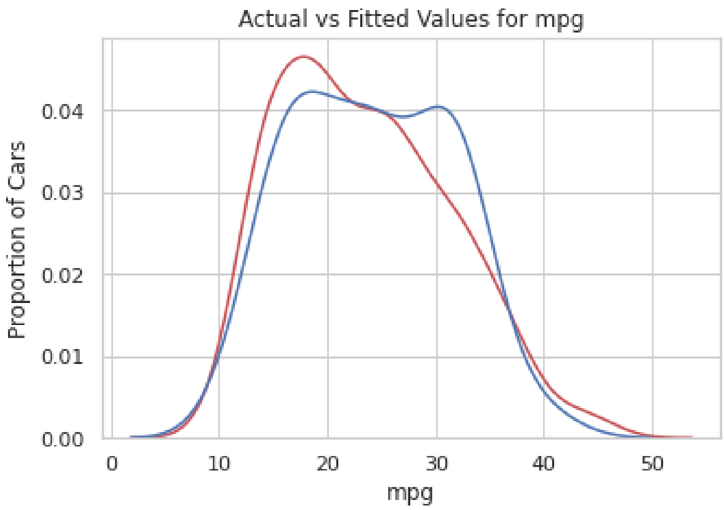
In [49]:

```
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred2, hist=False, color="b", label="Fitted Values" , ax=ax1)


plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```




**We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.**

In [50]:

```
from sklearn.metrics import r2_score,mean_squared_error
```

In [51]:

```
r2_score(y_test,y_pred2)
```

Out[51]:

```
0.9013457876319049
```

In [52]:

```
mean_squared_error(y_test,y_pred2)
```

```
mean_squared_error(y_test,y_pred2)
```

Out[52]:

```
6.818917500000002
```

In [53]:

```
np.sqrt(mean_squared_error(y_test,y_pred2))
```

Out[53]:

```
2.611305707878724
```

# linear regression

In [54]:

```
from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
```

Out[54]:

```
LinearRegression()
```

In [55]:

```
y_pred3=mr.predict(x_test)
y_pred3
```

Out[55]:

```
array([[13.20818031],
       [24.27993342],
       [11.61339788],
       [20.96914745],
       [17.7247275 ],
       [29.44595217],
       [33.47372984],
       [23.1855594 ],
       [15.045202  ],
       [26.79998444],
       [32.32754229],
       [33.93400668],
       [21.48572281],
       [25.80404696],
       [16.32002867],
       [30.62069212],
       [28.3611479 ],
       [28.68598061],
       [17.66367225],
       [31.02921296],
       [15.54781059],
       [24.61489613],
       [26.90655487],
       [20.51716586],
       [29.66216351],
       [28.48379869],
       [31.00137585],
       [29.9752557 ],
       [29.90123742],
       [18.07465439],
       [20.36226872],
       [31.32907003],
       [20.95979818],
       [32.03796407],
       [23.8731354 ],
       [26.30724058],
       [21.37158555],
       [16.80870416],
       [32.14991802],
```

```
       [ 9.27600756]])
```

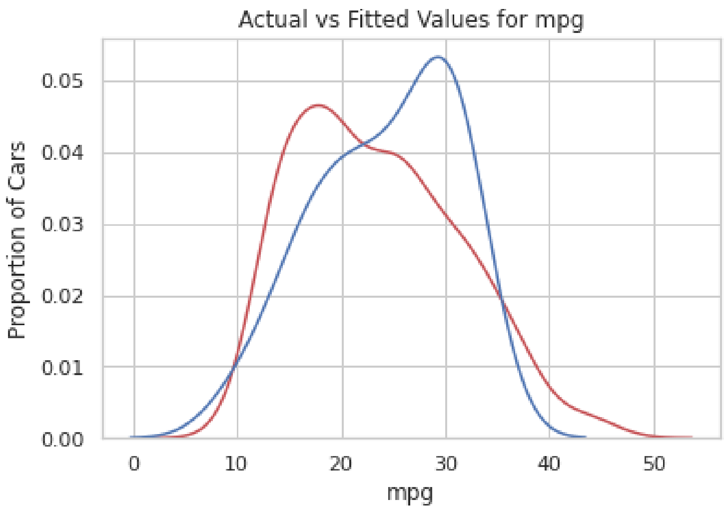
In [56]:

```
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r", label="Actual Value")
sns.distplot(y_pred3, hist=False, color="b", label="Fitted Values" , ax=ax1)


plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```




**We can see that the fitted values are not as close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.**

In [57]:

```
from sklearn.metrics import r2_score,mean_squared_error
r2_score(y_test,y_pred3)
```

Out[57]:

```
0.8460443802521529
```

In [58]:

```
mean_squared_error(y_test,y_pred3)
```

Out[58]:

```
10.64131621470885
```

In [59]:

```
np.sqrt(mean_squared_error(y_test,y_pred3))
```

Out[59]:

```
3.262103035575187
```

**Conclusion:**

**When comparing models, the model with the higher R-squared value is a better fit for the data.**

**When comparing models, the model with the smallest MSE value is a better fit for the data.**

**Comparing these three models, we conclude that the DecisionTree model is the best model to be able to predict mpg from our dataset.**