Auto_Milage Project

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

This dataset is a slightly modified version of the dataset provided in the StatLib library. In line with the use by Ross Quinlan (1993) in predicting the attribute "mpg", 8 of the original instances were removed because they had unknown values for the "mpg" attribute. The original dataset is available in the file "auto-mpg.data-original".

"The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993)

Lets get started!

Loading the data

In [17]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [98]:
```
df = pd.read_csv('auto-mpg.data', names=['milage','clyn','disp','hp','wt','ac
c','model','origin','name'])
```

Data Check

Check if any kind of data processing required

In [99]:
```
df.head()
```

Out[99]:

	milage	clyn	disp	hp	wt	acc	model	origin	name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

Data Cleaning and Pre-Processing

Dropping any null value. Here I am dropping null values because I have checked and these null values are not in significant numbers

In [100]:
```
df.dropna(inplace=True)
```

In [102]:
```
df.isnull().T.any().T.any()
```

Out[102]:
```
False
```

In [103]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 396 entries, 0 to 397
Data columns (total 9 columns):
milage    396 non-null float64
clyn      396 non-null int64
disp      396 non-null float64
hp        396 non-null object
wt        396 non-null float64
acc       396 non-null float64
model     396 non-null int64
origin    396 non-null object
name      396 non-null object
dtypes: float64(4), int64(2), object(3)
memory usage: 30.9+ KB
```

Dropping the Name field; this field does not have any significant impact on prediction

In [104]:
```
df1 = df.drop('name', axis=1)
```

In [105]:
```
df1.head()
```

Out[105]:

	milage	clyn	disp	hp	wt	acc	model	origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	1

In [106]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 396 entries, 0 to 397
Data columns (total 8 columns):
milage    396 non-null float64
clyn      396 non-null int64
disp      396 non-null float64
hp        396 non-null object
wt        396 non-null float64
acc       396 non-null float64
model     396 non-null int64
origin    396 non-null object
dtypes: float64(4), int64(2), object(2)
memory usage: 27.8+ KB
```

Converting Object types in the dataframe to float type

In [107]:
```
df1 = df1.convert_objects(convert_numeric=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWar
ning: convert_objects is deprecated.  Use the data-type specific converters p
d.to_datetime, pd.to_timedelta and pd.to_numeric.
  """Entry point for launching an IPython kernel.
```

In [108]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 396 entries, 0 to 397
Data columns (total 8 columns):
milage    396 non-null float64
clyn      396 non-null int64
disp      396 non-null float64
hp        392 non-null float64
wt        396 non-null float64
acc       396 non-null float64
model     396 non-null int64
origin    396 non-null int64
dtypes: float64(5), int64(3)
memory usage: 27.8 KB
```

In [109]:
```
df1.describe()
```

Out[109]:

	milage	clyn	disp	hp	wt	acc	m
count	396.000000	396.000000	396.000000	392.000000	396.000000	396.000000	396.00(
mean	23.522222	5.457071	193.516414	104.469388	2970.502525	15.552020	76.000(
std	7.834679	1.703511	104.511118	38.491160	848.963173	2.752512	3.6933(
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000(
25%	17.375000	4.000000	103.250000	75.000000	2222.250000	13.800000	73.000(
50%	23.000000	4.000000	146.000000	93.500000	2797.500000	15.500000	76.000(
75%	29.000000	8.000000	263.250000	126.000000	3610.000000	17.125000	79.000(
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000(

Final Null check and deleting any values if present

In [110]:
```
df1.isnull().any()
```

Out[110]:
```
milage    False
clyn      False
disp      False
hp         True
wt        False
acc       False
model     False
origin    False
dtype: bool
```

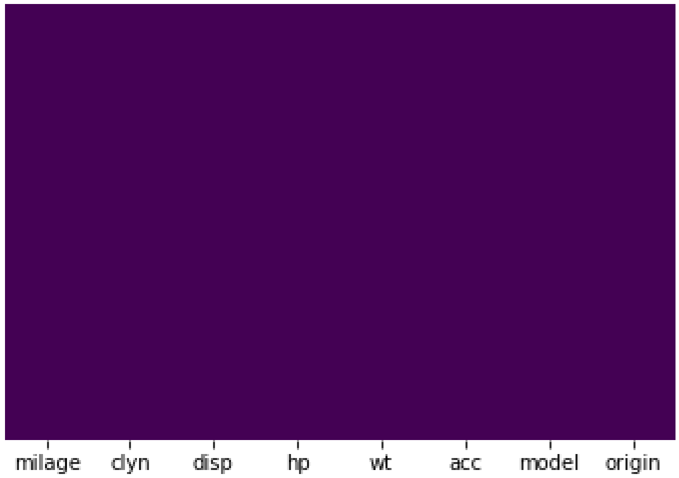
In [112]:
```
df1.dropna(inplace=True)
```

In [113]:
```
df1.isnull().T.any().T.any()
```

Out[113]:
```
False
```

In [116]:
```
sns.heatmap(df1.isnull(), yticklabels=False, cmap='viridis', cbar=False)
```

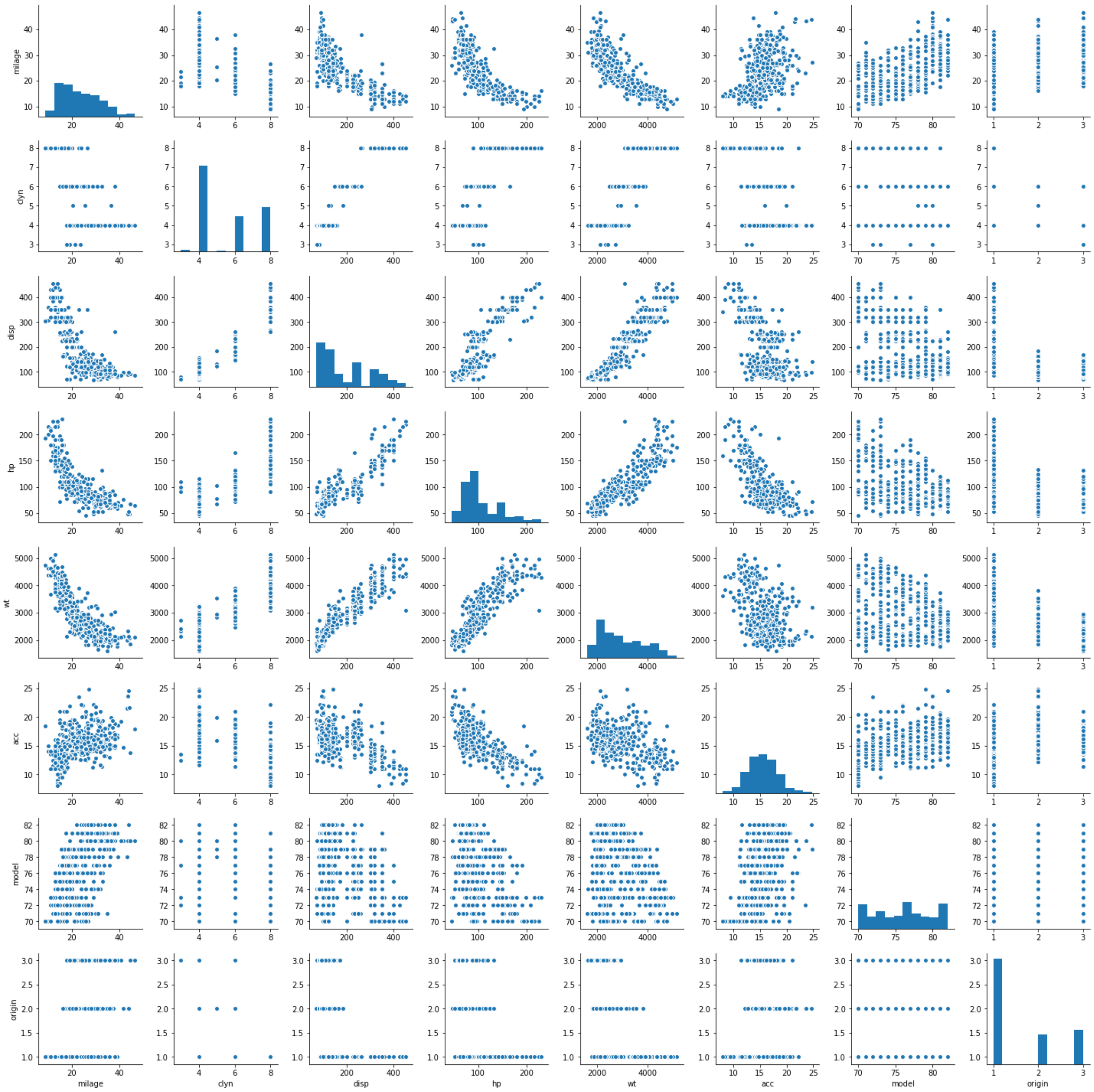
Out[116]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x84807f0>
```



Exploratory Data Analysis

In [117]:
```
sns.pairplot(df1)
```

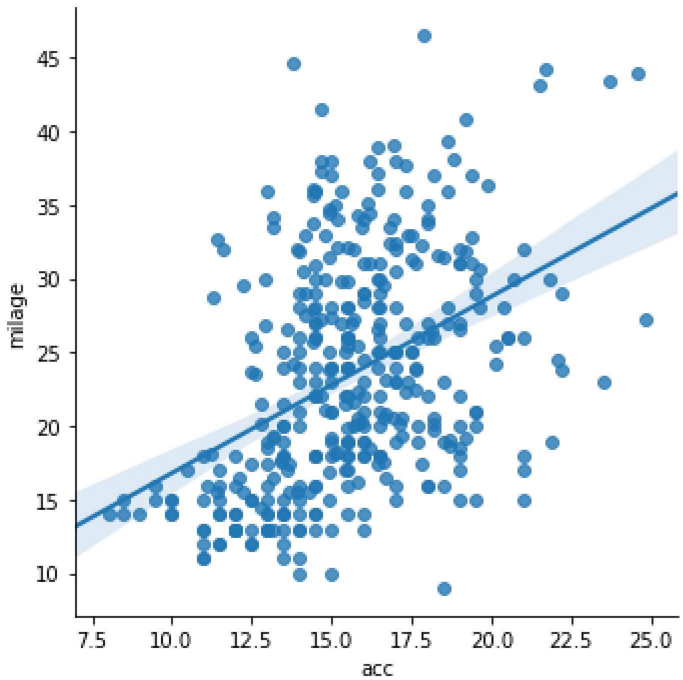
Out[117]:
```
<seaborn.axisgrid.PairGrid at 0xf341400>
```



Below are some of the significant and insignificant linear relationships

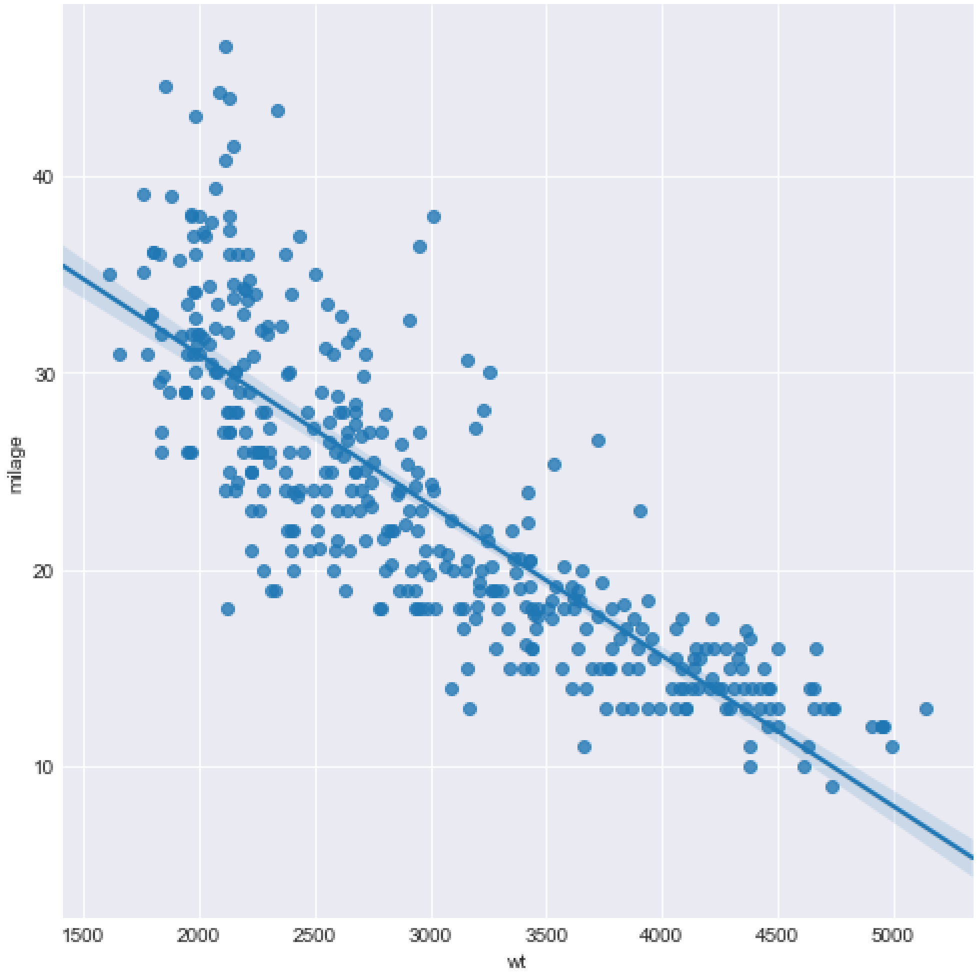
In [135]:
```
sns.lmplot('acc', 'milage',df1)
```

Out[135]: <seaborn.axisgrid.FacetGrid at 0xee28f60>



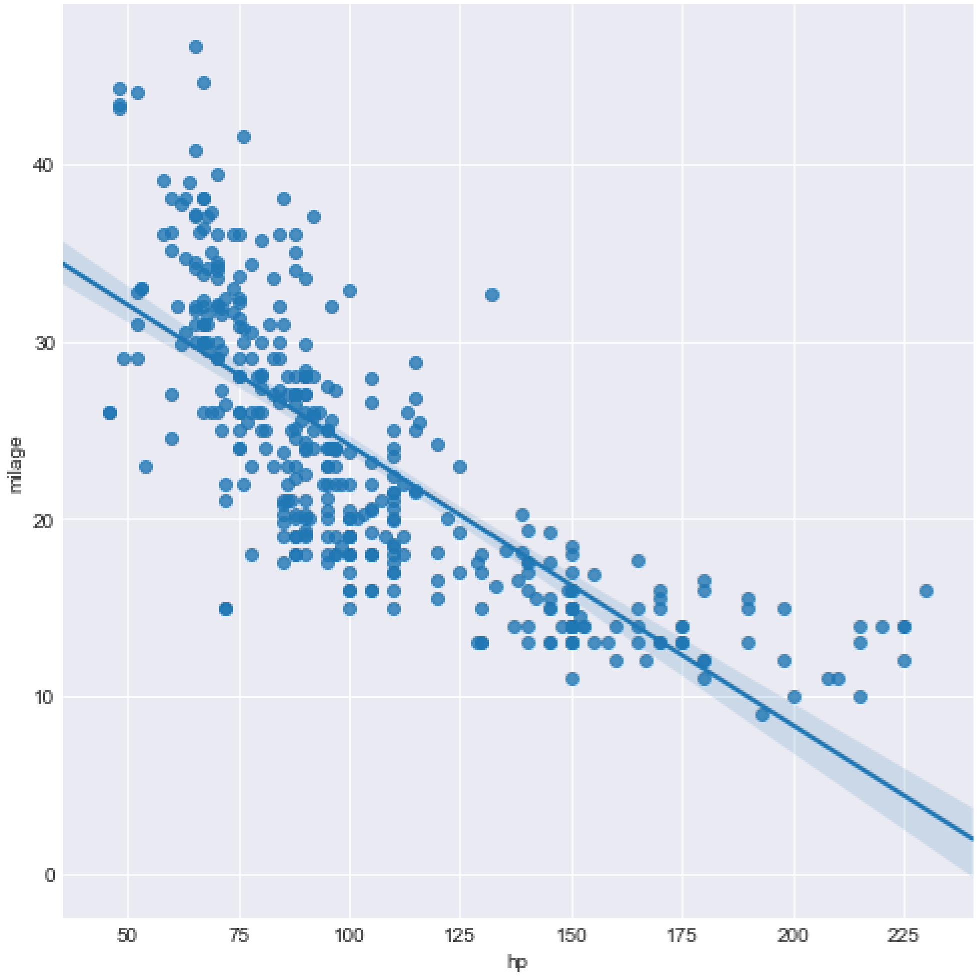
In [151]:
```
sns.lmplot(x='wt',y='milage',data=df1,size=7)
```

Out[151]: <seaborn.axisgrid.FacetGrid at 0x169ad2b0>



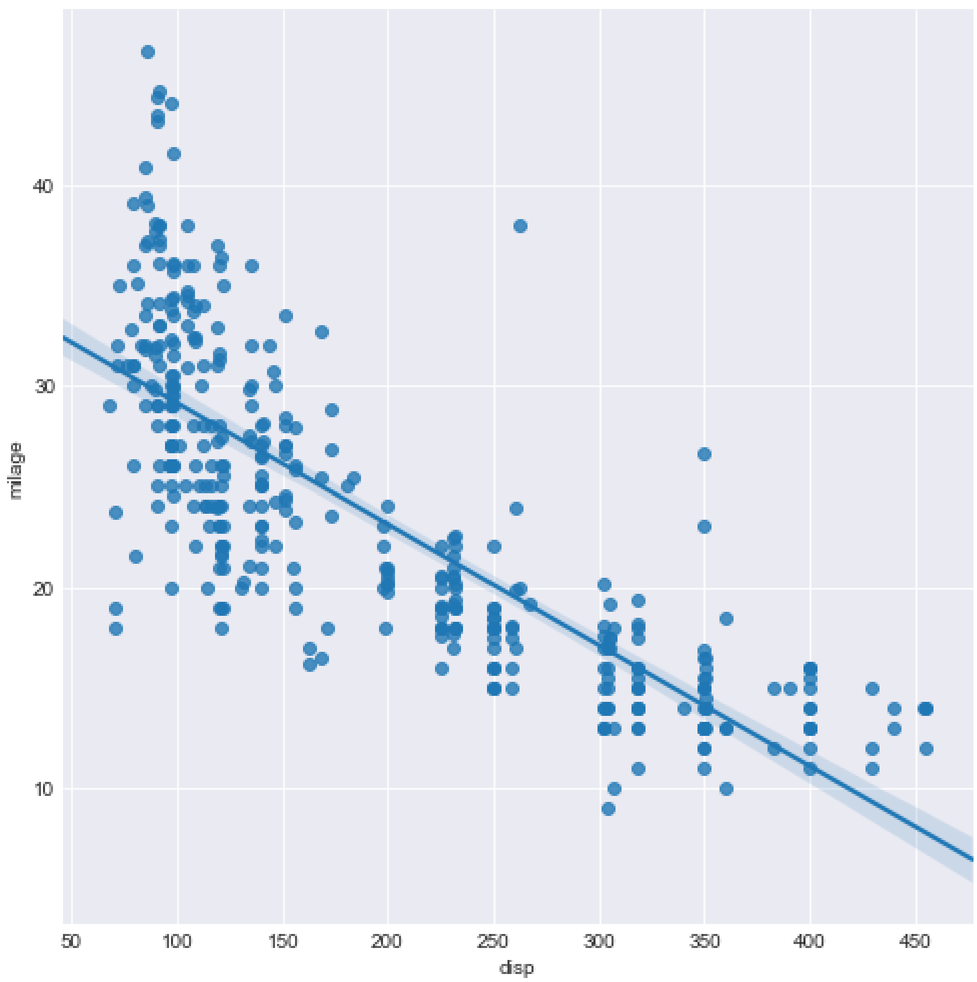
In [152]:
```
sns.lmplot(x='hp',y='milage',data=df1, size=7)
```

Out[152]: <seaborn.axisgrid.FacetGrid at 0x10e4ff60>



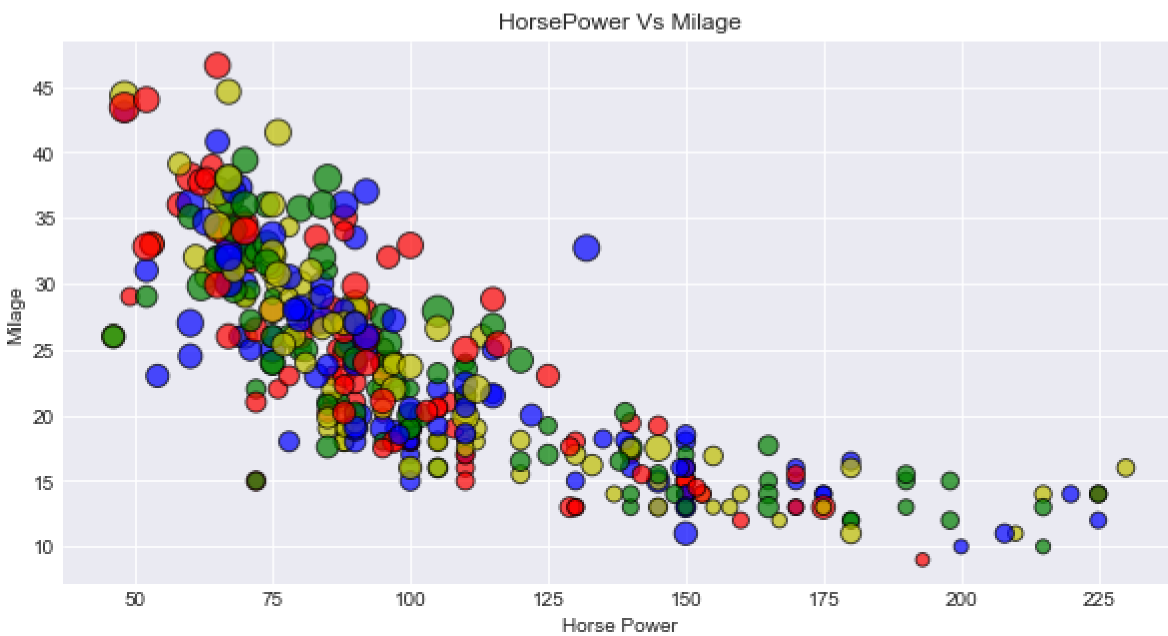
In [153]:
```
sns.lmplot(x='disp',y='milage',data=df1, size=7)
```

Out[153]: <seaborn.axisgrid.FacetGrid at 0x13849fd0>



In [145]:
```
plt.figure(figsize=(10,5))
plt.title('HorsePower Vs Milage')
plt.xlabel('Horse Power')
plt.ylabel('Milage')
plt.style.use('seaborn-darkgrid')
plt.scatter(df1['hp'], df1['milage'], alpha=0.7,marker='o', s=df.milage*5, c=[
'r','g','b','y'], edgecolors='black')
```

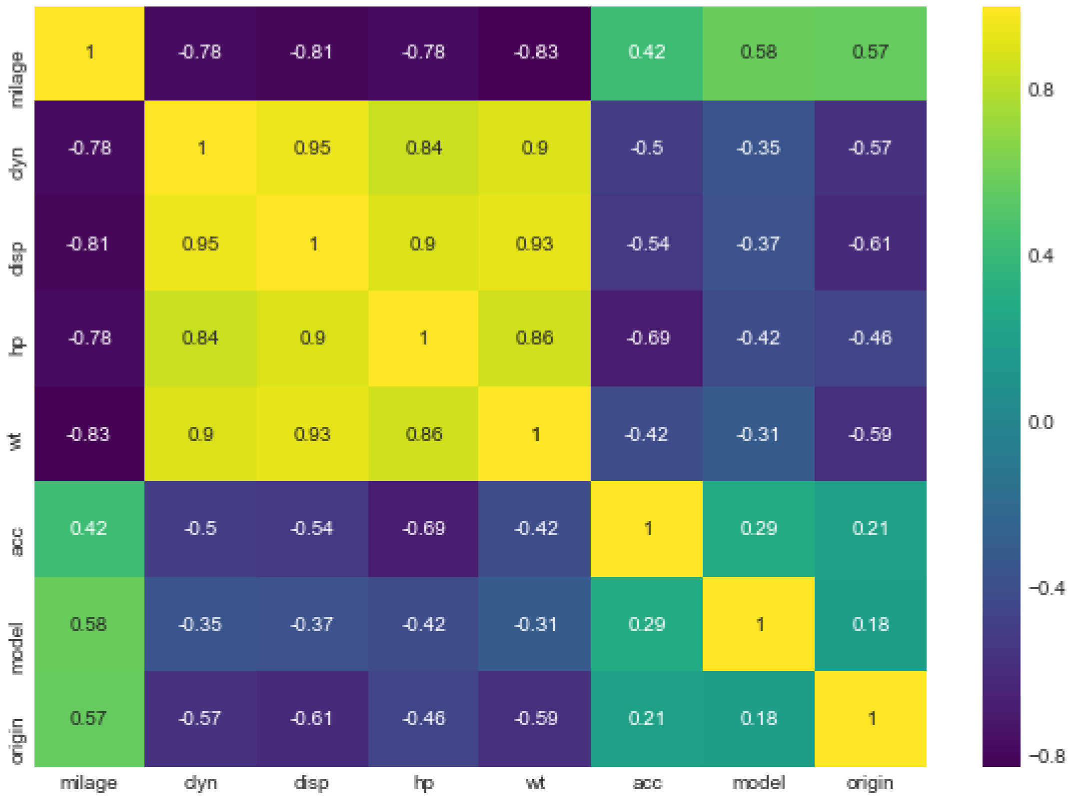
Out[145]: <matplotlib.collections.PathCollection at 0x16b2ca90>



Correlation Metrics with HeatMap

In [158]:
```
plt.subplots(figsize=(10,7))
sns.heatmap(df1.corr(), annot=True, cmap='viridis')
```

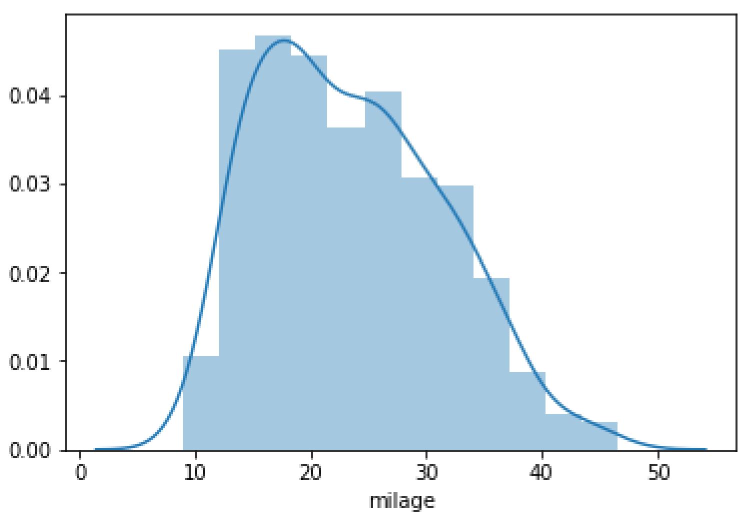
Out[158]: <matplotlib.axes._subplots.AxesSubplot at 0x1100e518>



Lets check how the dependent variable is distributed

In [118]:
```
sns.distplot(df.milage)
```

Out[118]: <matplotlib.axes._subplots.AxesSubplot at 0x160c61d0>



Training and Testing Data

Now that we've explored the data a bit, let's go ahead and split the data into training and testing sets.

In [159]:
```
from sklearn.model_selection import train_test_split
```

In [166]:
```
X_train, X_test, y_train, y_test = train_test_split(df1.drop('milage', axis=1
), df1['milage'], test_size=0.3, random_state=101)
```

Training the Model

Now its time to train our model on our training data!

Import LinearRegression from sklearn.linear_model

In [167]:
```
from sklearn.linear_model import LinearRegression
```

In [168]:
```
lm = LinearRegression()
```

In [169]:
```
lm.fit(X_train, y_train)
```

Out[169]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Predicting the Variables

In [170]:
```
pred = lm.predict(X_test)
```

Evaluating the Model

Now evaluating the model with r2 score and coefficients and errors

R2 Score

In [171]:
```
from sklearn import metrics
```

In [172]:
```
print(metrics.r2_score(y_test, pred))
```

```
0.803256874483
```

Coefficients

In [179]:
```
coefficients = pd.DataFrame(data=lm.coef_, index=df1.columns[1:], columns=['Co
efficients'])
```

In [180]:
```
coefficients
```

Out[180]:

	Coefficients
clyn	-0.215449
disp	0.016572
hp	0.003190
wt	-0.007320
acc	0.238678
model	0.808829
origin	1.158099

Interpreting the coefficients:

- Holding all other features fixed, 1 unit increase in milage is associated with **decrease of 0.215449 unit in cylinder**.
- Holding all other features fixed, 1 unit increase in milage is associated with **increase of 0.016572 unit in displacement**.
- Holding all other features fixed, 1 unit increase in milage is associated with **increase of 0.003190 unit in Horsepower**.
- Holding all other features fixed, 1 unit increase in milage is associated with **decrease of 0.007320 unit in Weight**.
- Holding all other features fixed, 1 unit increase in milage is associated with **increase of 0.238678 unit in acceleration**.
- Holding all other features fixed, 1 unit increase in milage is associated with **increase of 0.808829 unit in year**.
- Holding all other features fixed, 1 unit increase in milage is associated with **increase of 1.158099 unit in origin**.

Calculating Errors

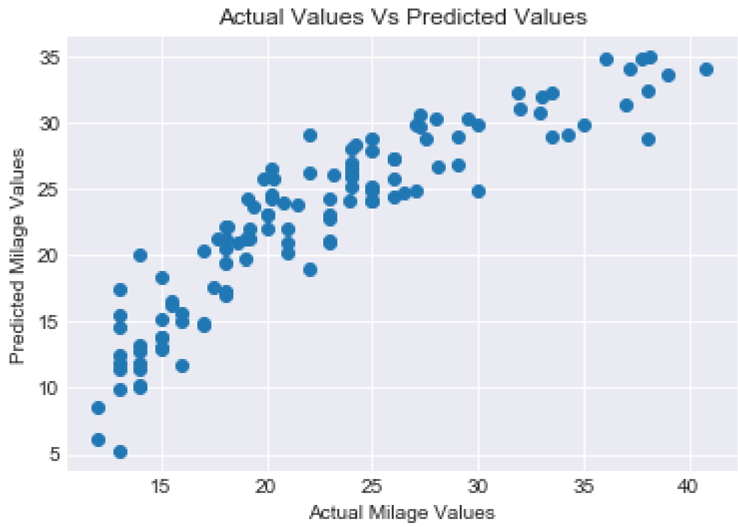
In [181]:
```
print('MAE: ',metrics.mean_absolute_error(y_test, pred))
print('MSE: ', metrics.mean_squared_error(y_test, pred))
print('RMSE: ', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE:  2.57842905838
MSE:  10.1598727925
RMSE:  3.18745553577
```

Plotting the Test Values Vs Predicted Values

In [184]:
```
plt.title('Actual Values Vs Predicted Values')
plt.xlabel('Actual Milage Values')
plt.ylabel('Predicted Milage Values')
plt.scatter(y_test, pred)
```

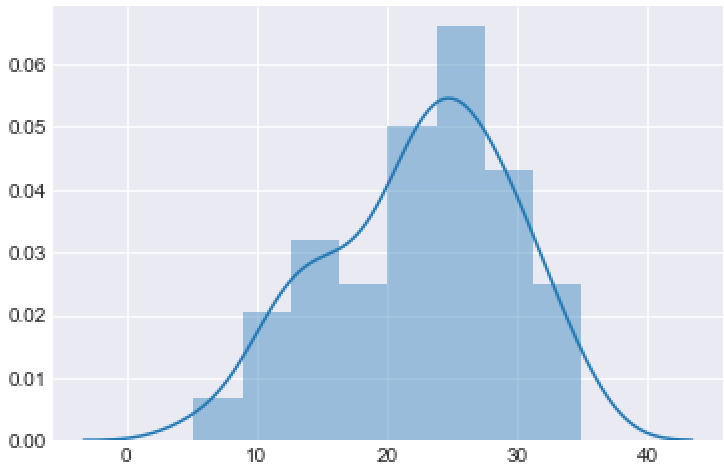
Out[184]: <matplotlib.collections.PathCollection at 0xfcebf28>



Distribution of Predicted Values

In [189]:
```
sns.distplot(pred)
```

Out[189]: <matplotlib.axes._subplots.AxesSubplot at 0x137deba8>



Residuals

You should have gotten a model with an good fit. Let's quickly explore the residuals to make sure everything was okay with our data.

Plot a histogram of the residuals and make sure it looks normally distributed. Use either seaborn distplot, or just plt.hist().

In [190]:
```
sns.distplot(y_test-pred)
```

Out[190]: <matplotlib.axes._subplots.AxesSubplot at 0x13736160>