Regression Models Course Project

August 20, 2020

You work for Motor Trend, a magazine about the automobile industry. Looking at a data set of a collection of cars, they are interested in exploring the relationship between a set of variables and miles per gallon (MPG) (outcome). They are particularly interested in the following two questions:

“Is an automatic or manual transmission better for MPG”

“Quantify the MPG difference between automatic and manual transmissions”

Take the mtcars data set and write up an analysis to answer their question using regression models and exploratory data analyses.

A data frame with 32 observations on 11 (numeric) variables.

mpg - Miles/(US) gallon

cyl - Number of cylinders

disp - Displacement (cu.in.)

hp - Gross horsepower

drat - Rear axle ratio

wt - Weight (1000 lbs)

qsec - 1/4 mile time

vs - Engine (0 = V-shaped, 1 = straight)

am - Transmission (0 = automatic, 1 = manual)

gear - Number of forward gears

carb - Number of carburetors

0.0.1 Import Libraries

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

import xgboost as xgb
from xgboost import XGBClassifier, XGBRegressor
from xgboost import to_graphviz, plot_importance

#from sklearn.experimental import enable_hist_gradient_boosting
#from sklearn.ensemble import _hist_gradient_boosting
#from sklearn.ensemble import HistGradientBoostingRegressor,␣
 ↪HistGradientBoostingRegressor

%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

from sklearn.model_selection import cross_val_score, train_test_split,␣
 ↪GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler,␣
 ↪OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪mean_absolute_error, mean_squared_error,r2_score
from sklearn.metrics import plot_confusion_matrix, plot_precision_recall_curve,␣
 ↪plot_roc_curve, accuracy_score
from sklearn.metrics import auc, f1_score, precision_score, recall_score,␣
 ↪roc_auc_score

import warnings
warnings.filterwarnings('ignore')

import pickle
from pickle import dump, load

np.random.seed(0)

#from pycaret.classification import *
#from pycaret.clustering import *
from pycaret.regression import *

pd.set_option('display.max_columns',100)
#pd.set_option('display.max_rows',100)
pd.set_option('display.width', 1000)
```

0.0.2 Data Exploration and Analysis

[2]:
```
df = pd.read_csv("mtcars.csv")
```

[3]:
```
df
```

[3]:
```
                    model   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear
carb
```

```
0             Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
4
1         Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
4
2            Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
1
3        Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
1
4     Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0     3
2
5               Valiant  18.1    6  225.0  105  2.76  3.460  20.22   1   0     3
1
6            Duster 360  14.3    8  360.0  245  3.21  3.570  15.84   0   0     3
4
7             Merc 240D  24.4    4  146.7   62  3.69  3.190  20.00   1   0     4
2
8              Merc 230  22.8    4  140.8   95  3.92  3.150  22.90   1   0     4
2
9              Merc 280  19.2    6  167.6  123  3.92  3.440  18.30   1   0     4
4
10            Merc 280C  17.8    6  167.6  123  3.92  3.440  18.90   1   0     4
4
11           Merc 450SE  16.4    8  275.8  180  3.07  4.070  17.40   0   0     3
3
12           Merc 450SL  17.3    8  275.8  180  3.07  3.730  17.60   0   0     3
3
13          Merc 450SLC  15.2    8  275.8  180  3.07  3.780  18.00   0   0     3
3
14   Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250  17.98   0   0     3
4
15  Lincoln Continental  10.4    8  460.0  215  3.00  5.424  17.82   0   0     3
4
16    Chrysler Imperial  14.7    8  440.0  230  3.23  5.345  17.42   0   0     3
4
17             Fiat 128  32.4    4   78.7   66  4.08  2.200  19.47   1   1     4
1
18          Honda Civic  30.4    4   75.7   52  4.93  1.615  18.52   1   1     4
2
19       Toyota Corolla  33.9    4   71.1   65  4.22  1.835  19.90   1   1     4
1
20        Toyota Corona  21.5    4  120.1   97  3.70  2.465  20.01   1   0     3
1
21     Dodge Challenger  15.5    8  318.0  150  2.76  3.520  16.87   0   0     3
2
22          AMC Javelin  15.2    8  304.0  150  3.15  3.435  17.30   0   0     3
2
23           Camaro Z28  13.3    8  350.0  245  3.73  3.840  15.41   0   0     3
```

```
4
24     Pontiac Firebird  19.2    8  400.0  175  3.08  3.845  17.05   0   0     3
2
25            Fiat X1-9  27.3    4   79.0   66  4.08  1.935  18.90   1   1     4
1
26        Porsche 914-2  26.0    4  120.3   91  4.43  2.140  16.70   0   1     5
2
27         Lotus Europa  30.4    4   95.1  113  3.77  1.513  16.90   1   1     5
2
28       Ford Pantera L  15.8    8  351.0  264  4.22  3.170  14.50   0   1     5
4
29         Ferrari Dino  19.7    6  145.0  175  3.62  2.770  15.50   0   1     5
6
30        Maserati Bora  15.0    8  301.0  335  3.54  3.570  14.60   0   1     5
8
31           Volvo 142E  21.4    4  121.0  109  4.11  2.780  18.60   1   1     4
2
```

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   model   32 non-null     object
 1   mpg     32 non-null     float64
 2   cyl     32 non-null     int64
 3   disp    32 non-null     float64
 4   hp      32 non-null     int64
 5   drat    32 non-null     float64
 6   wt      32 non-null     float64
 7   qsec    32 non-null     float64
 8   vs      32 non-null     int64
 9   am      32 non-null     int64
 10  gear    32 non-null     int64
 11  carb    32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

[5]:

```
df.describe(include='all')
```

[5]:

```
            model        mpg        cyl       disp         hp       drat
wt       qsec         vs         am       gear      carb
count           32  32.000000  32.000000   32.000000   32.000000  32.000000
32.000000  32.000000  32.000000  32.000000  32.000000  32.0000
unique          32        NaN        NaN         NaN         NaN        NaN
```

```
NaN        NaN        NaN        NaN        NaN      NaN
top     Fiat X1-9        NaN        NaN         NaN         NaN        NaN
NaN        NaN        NaN        NaN        NaN      NaN
freq            1        NaN        NaN         NaN         NaN        NaN
NaN        NaN        NaN        NaN        NaN      NaN
mean          NaN  20.090625   6.187500  230.721875  146.687500   3.596563
3.217250  17.848750   0.437500   0.406250   3.687500   2.8125
std           NaN   6.026948   1.785922  123.938694   68.562868   0.534679
0.978457   1.786943   0.504016   0.498991   0.737804   1.6152
min           NaN  10.400000   4.000000   71.100000   52.000000   2.760000
1.513000  14.500000   0.000000   0.000000   3.000000   1.0000
25%           NaN  15.425000   4.000000  120.825000   96.500000   3.080000
2.581250  16.892500   0.000000   0.000000   3.000000   2.0000
50%           NaN  19.200000   6.000000  196.300000  123.000000   3.695000
3.325000  17.710000   0.000000   0.000000   4.000000   2.0000
75%           NaN  22.800000   8.000000  326.000000  180.000000   3.920000
3.610000  18.900000   1.000000   1.000000   4.000000   4.0000
max           NaN  33.900000   8.000000  472.000000  335.000000   4.930000
5.424000  22.900000   1.000000   1.000000   5.000000   8.0000
```

[6]:
```
df.shape
```

[6]:
```
(32, 12)
```

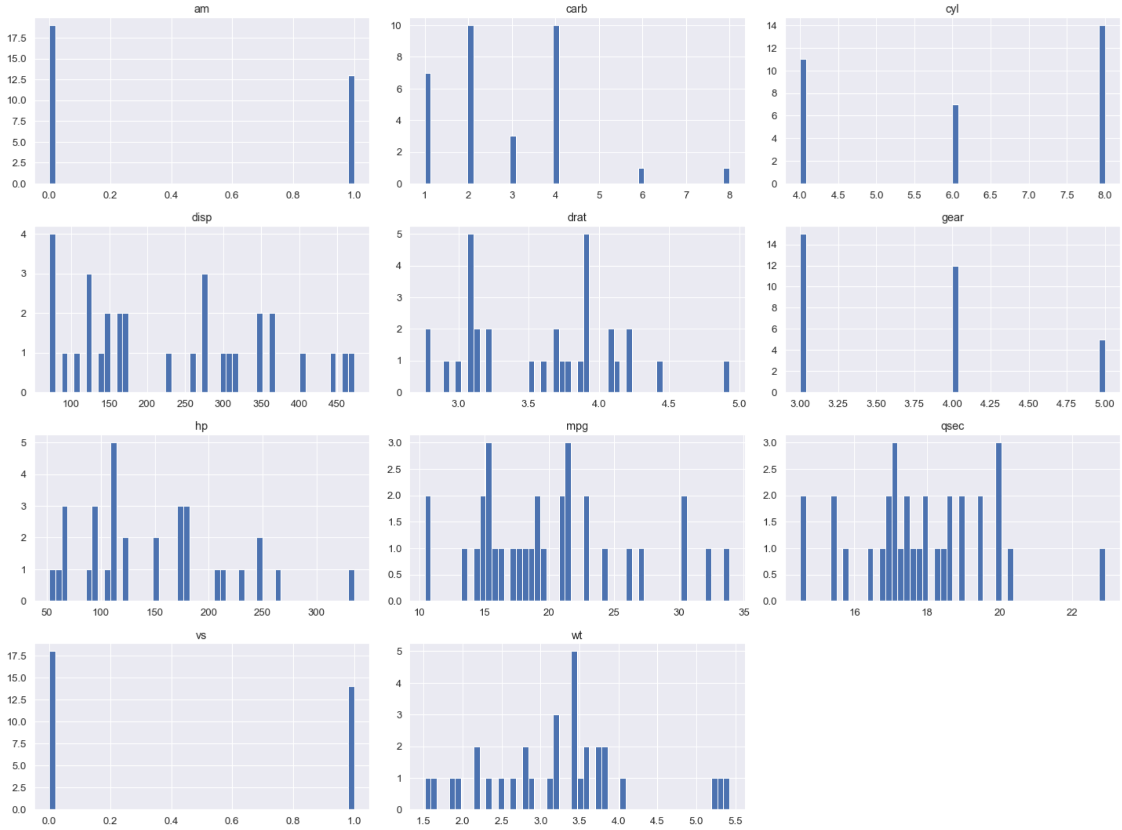
[7]:
```
df.columns
```

[7]:
```
Index(['model', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am',
'gear', 'carb'], dtype='object')
```

0.0.3 Data Visualization

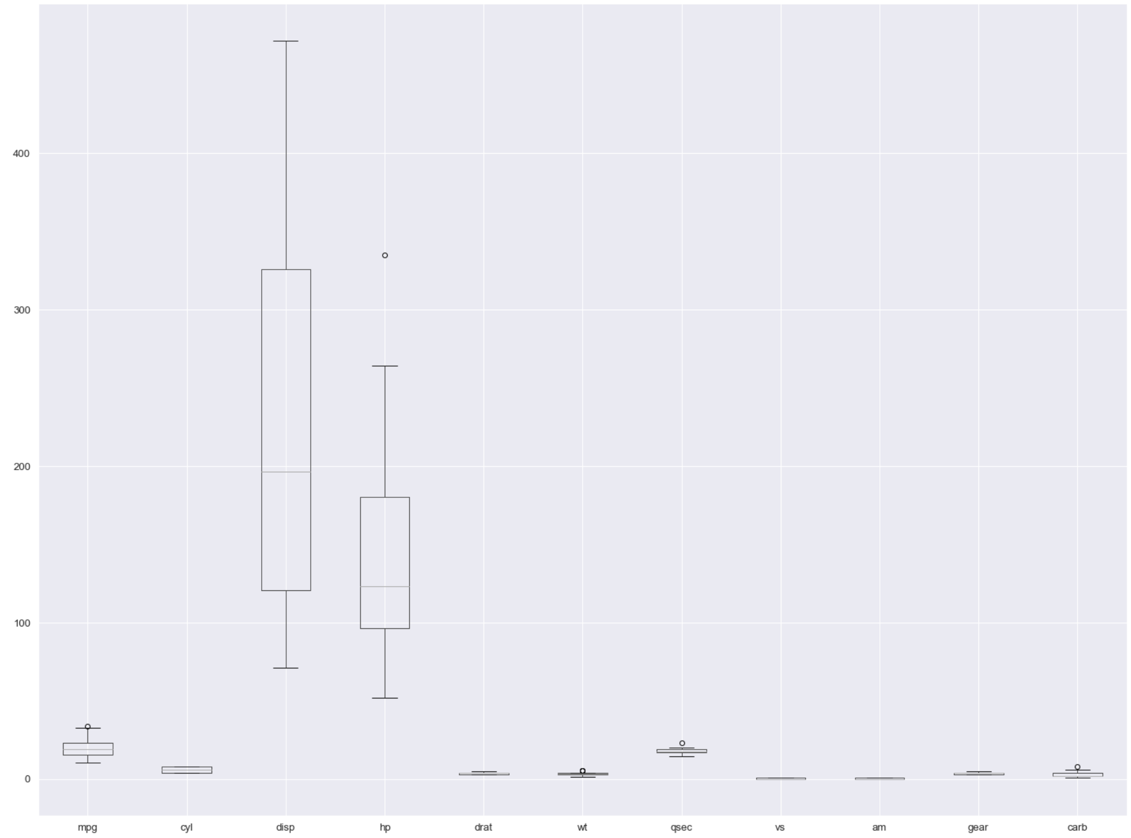
0.0.4 Univariate Data Exploration

[8]:
```
df.hist(bins=50, figsize=(20,15))
plt.tight_layout()
plt.show()
```



[9]:

```
df.boxplot(figsize=(20,15))
plt.tight_layout()
plt.show()
```

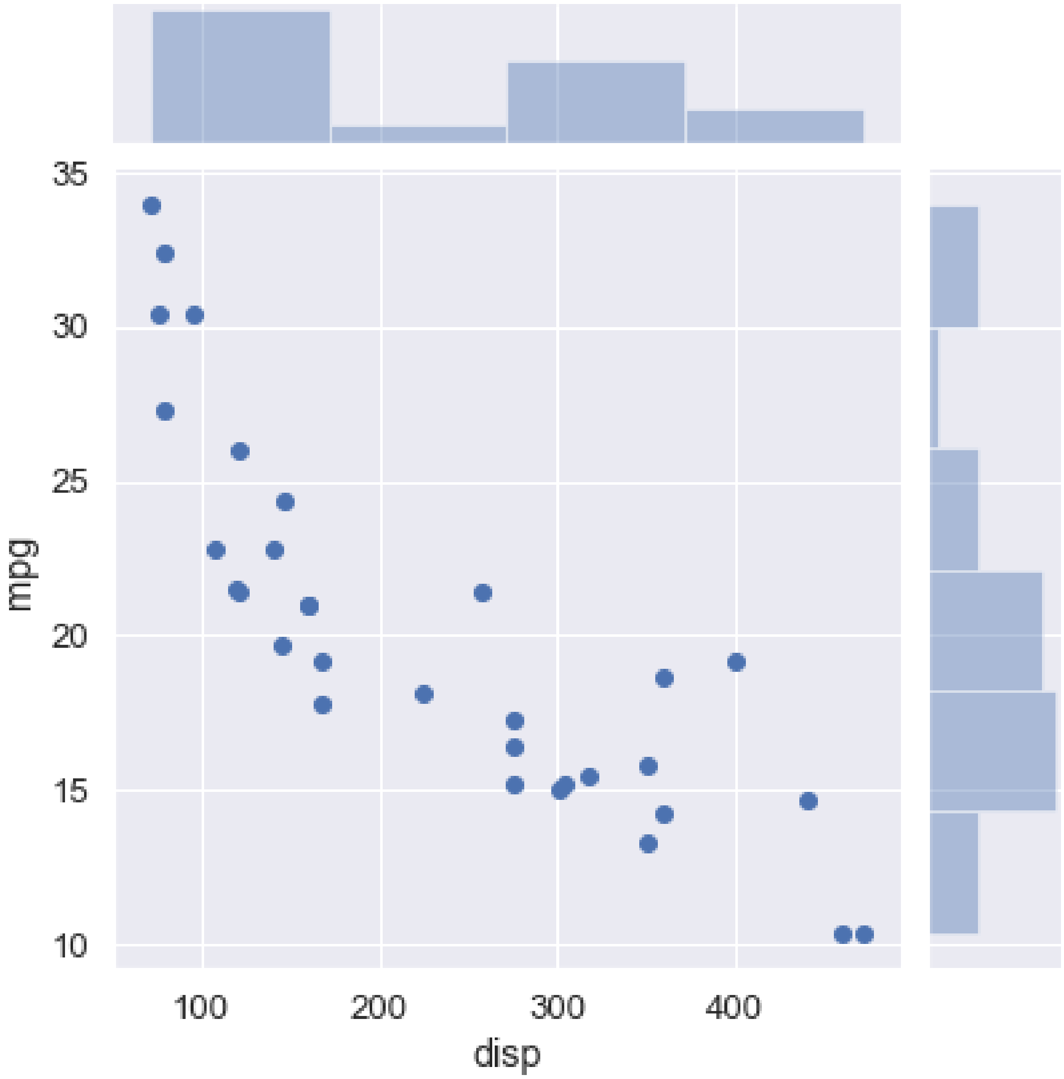


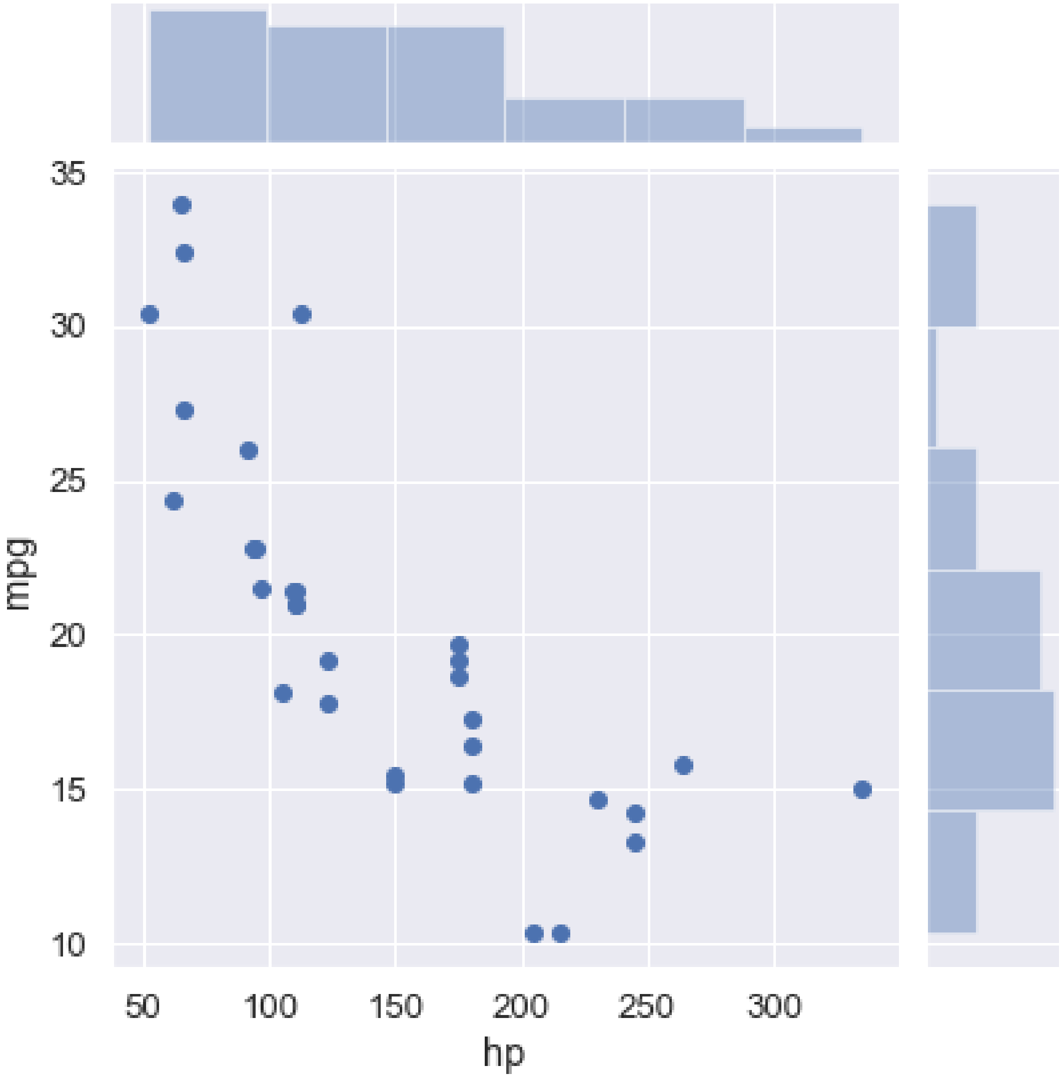
0.0.5 Bivariate Data Exploration

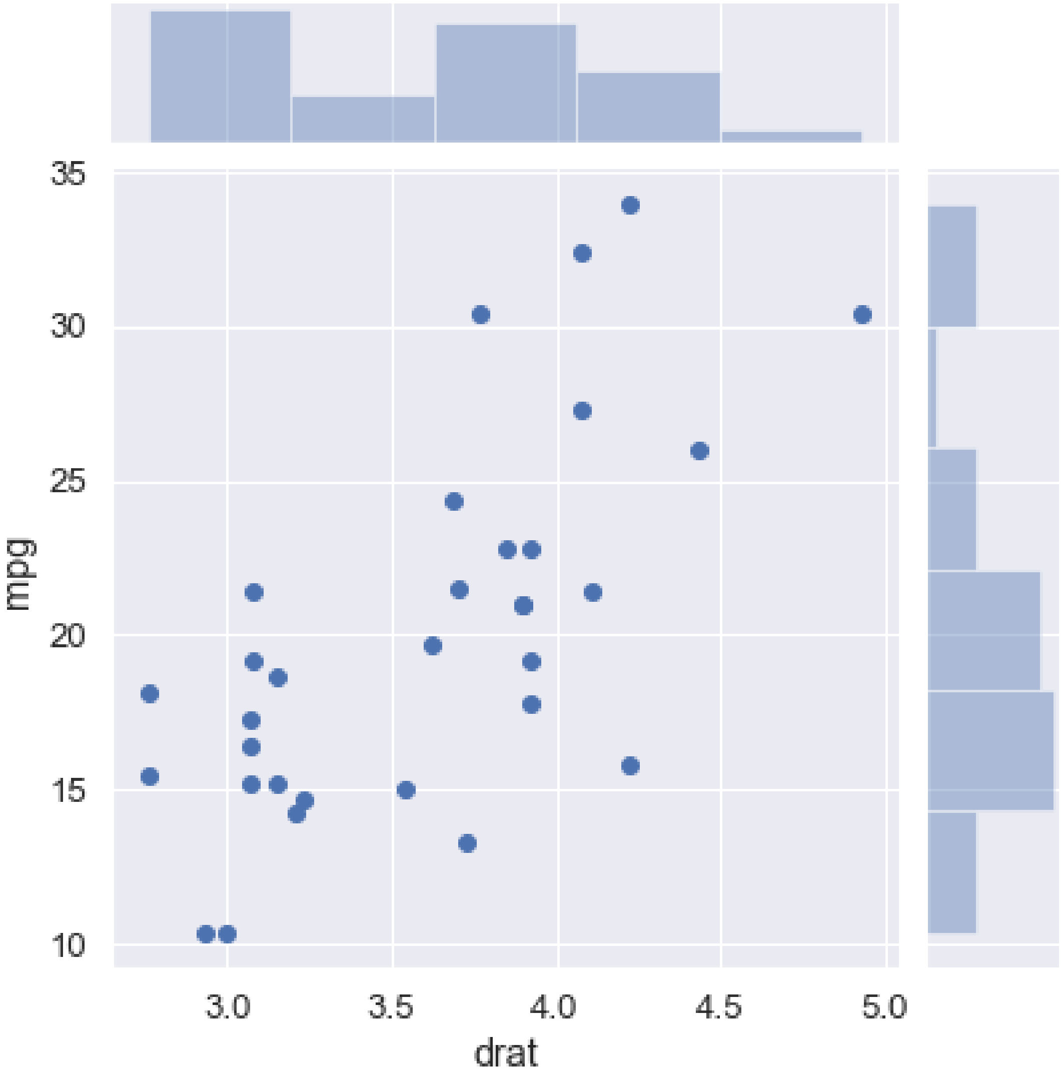
[10]:

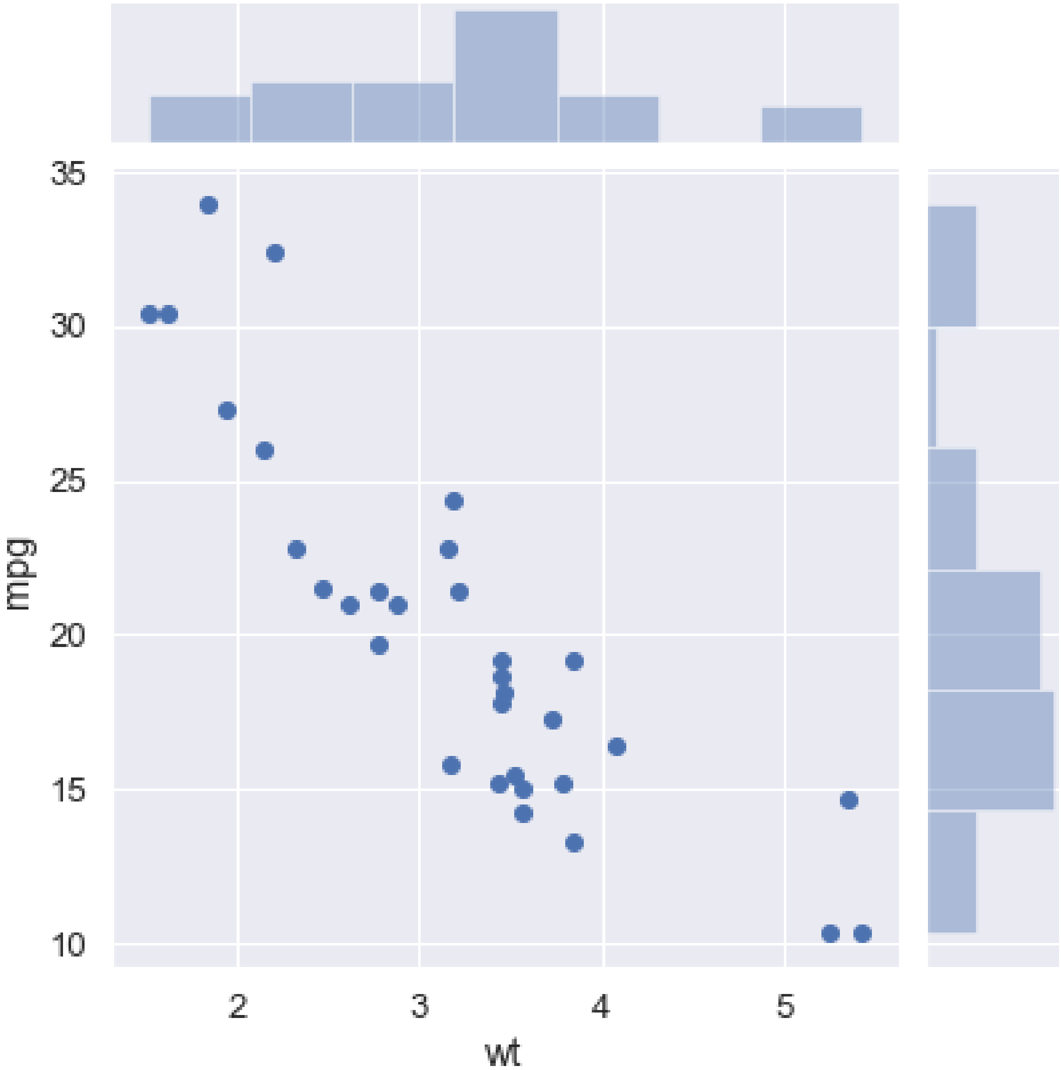
```python
sns.jointplot(x='disp', y='mpg',data=df, kind='scatter')

sns.jointplot(x='hp', y='mpg',data=df, kind='scatter')

sns.jointplot(x='drat', y='mpg',data=df, kind='scatter')

sns.jointplot(x='wt', y='mpg',data=df, kind='scatter')

sns.jointplot(x='qsec', y='mpg',data=df, kind='scatter')

sns.jointplot(x='vs', y='mpg',data=df, kind='scatter')

sns.jointplot(x='am', y='mpg',data=df, kind='scatter')

# sns.jointplot(x='', y='',data=df, kind='reg')

# sns.jointplot(x='', y='',data=df, kind='reg')
```

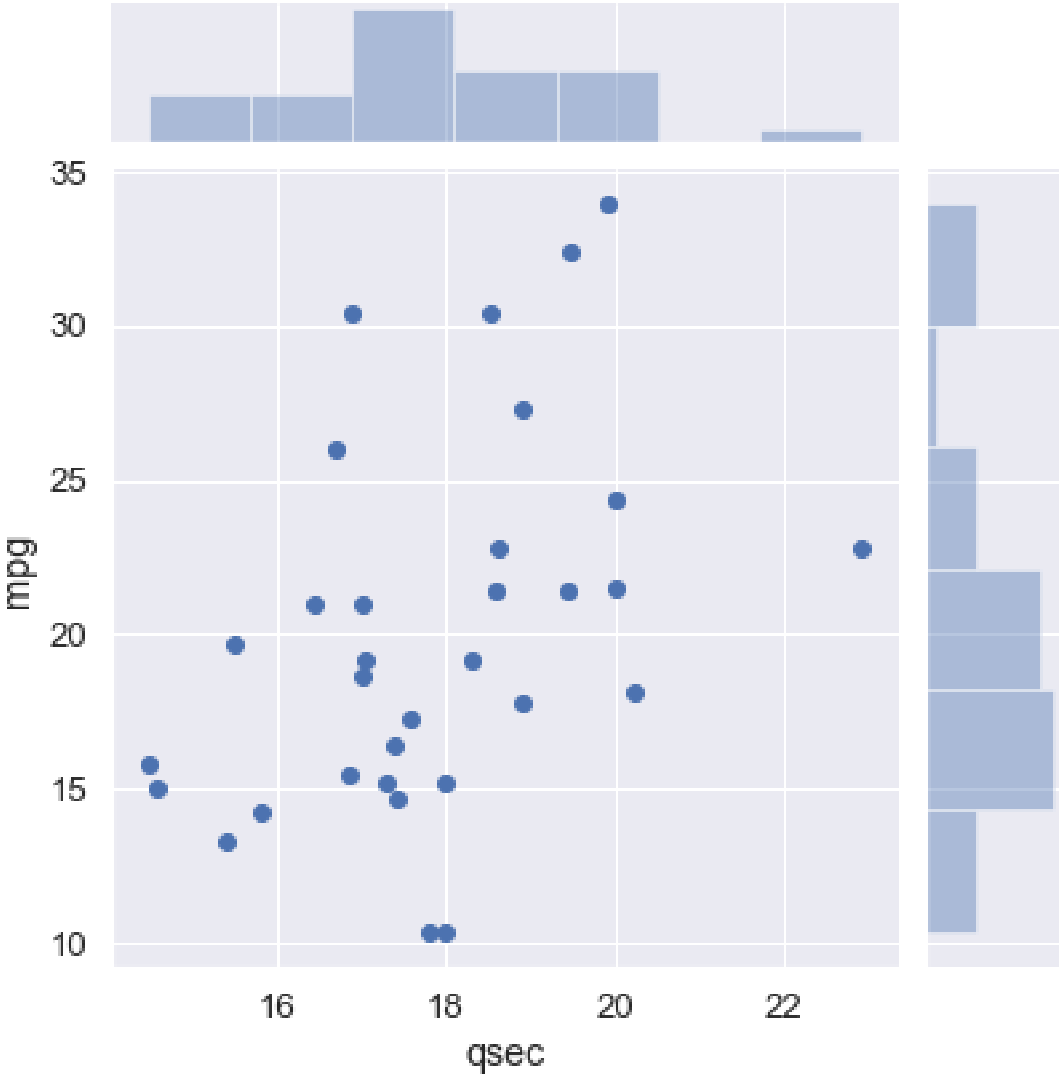
```
plt.show()
```

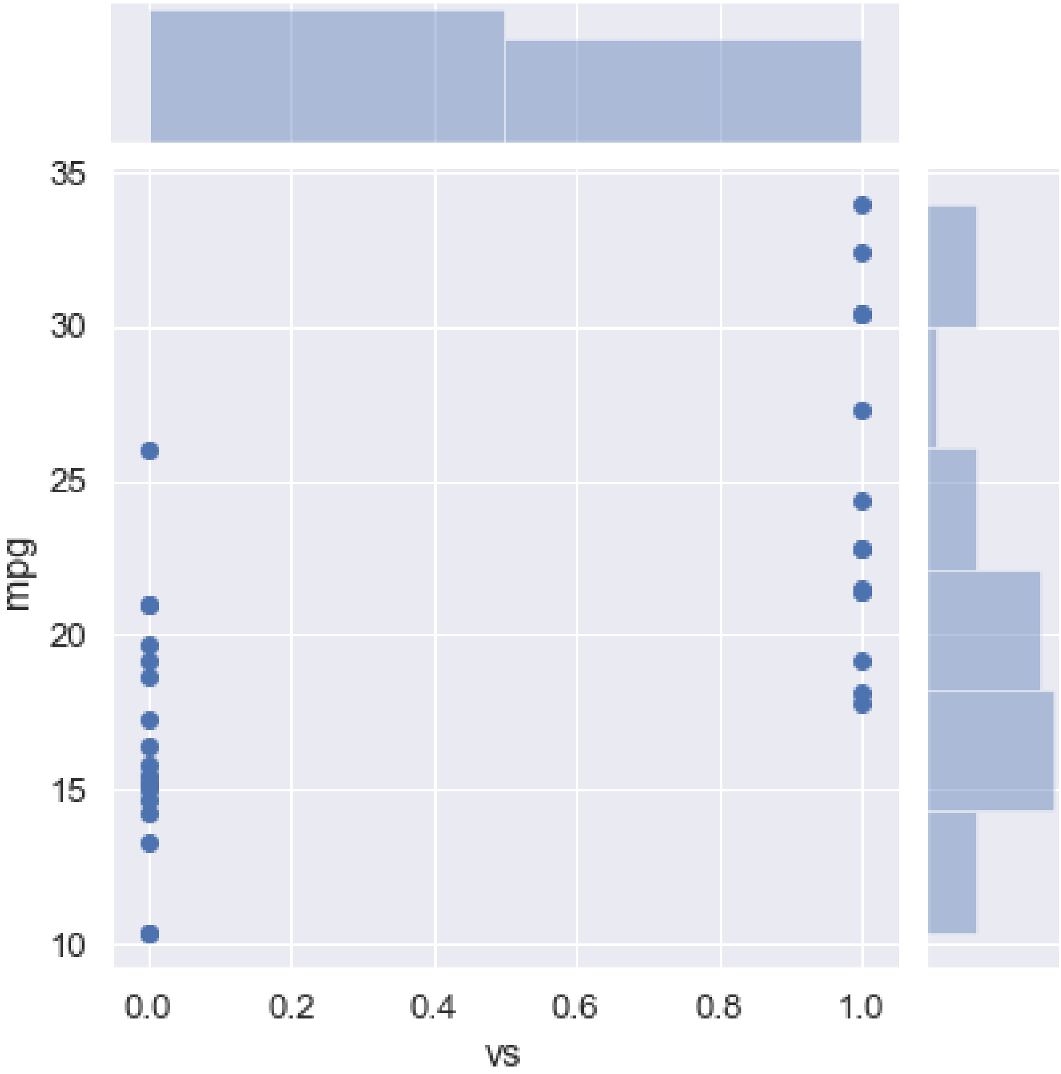


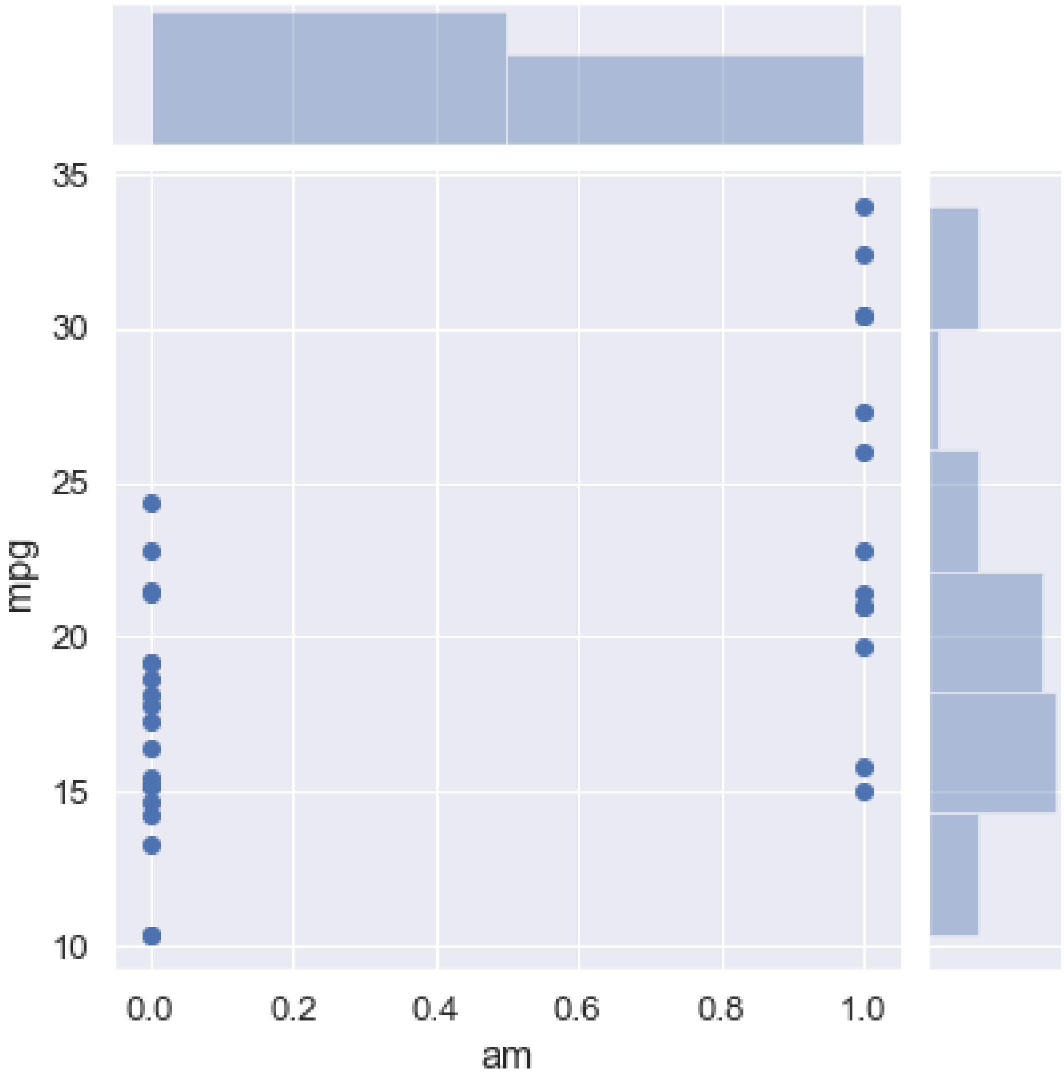

35
30
25
20
15
10
mpg
50
100
150
200
250
300
hp


35
30
25
20
15
10
mpg
3.0
3.5
4.0
4.5
5.0
drat


35
30
25
20
15
10
mpg
2
3
4
5
wt

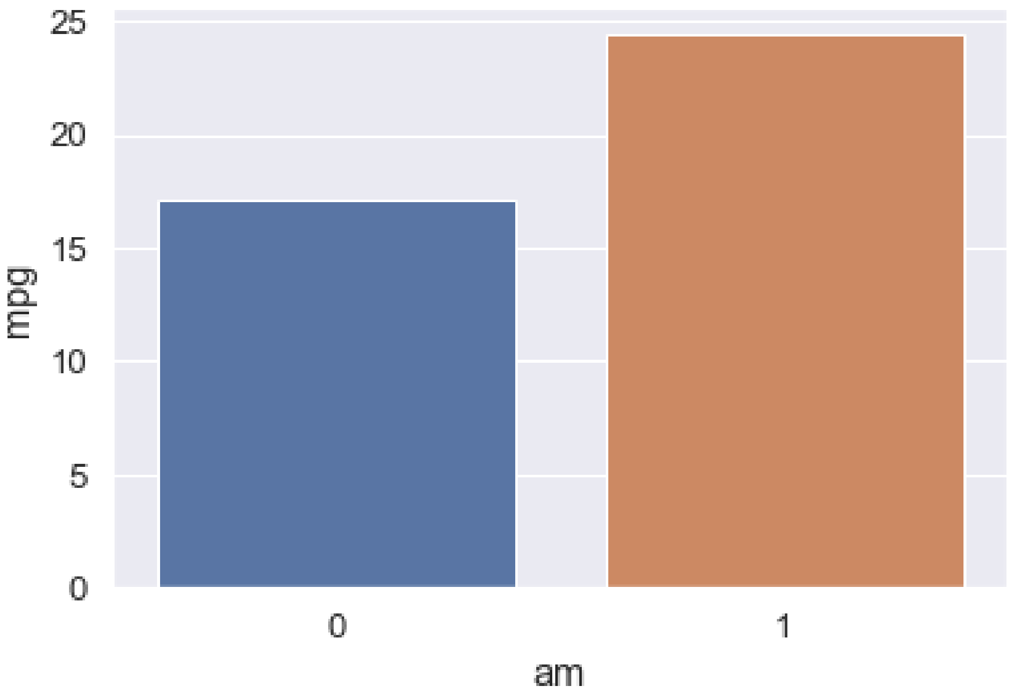

35
30
25
20
15
10
mpg
16
18
20
22
qsec


35
30
25
20
15
10
mpg
0.0
0.2
0.4
0.6
0.8
1.0
vs



[11]:

```
sns.barplot(x='am', y='mpg',data=df, ci=None);
```



Result: Manual gear gives higher MPG outcome by 10

```
[12]: df.groupby("am").mean()
```

```
[12]:           mpg       cyl        disp          hp      drat        wt       qsec
      vs       gear      carb
      am
      0   17.147368  6.947368  290.378947  160.263158  3.286316  3.768895  18.183158
      0.368421  3.210526  2.736842
      1   24.392308  5.076923  143.530769  126.846154  4.050000  2.411000  17.360000
      0.538462  4.384615  2.923077
```

0.0.6 Correlation

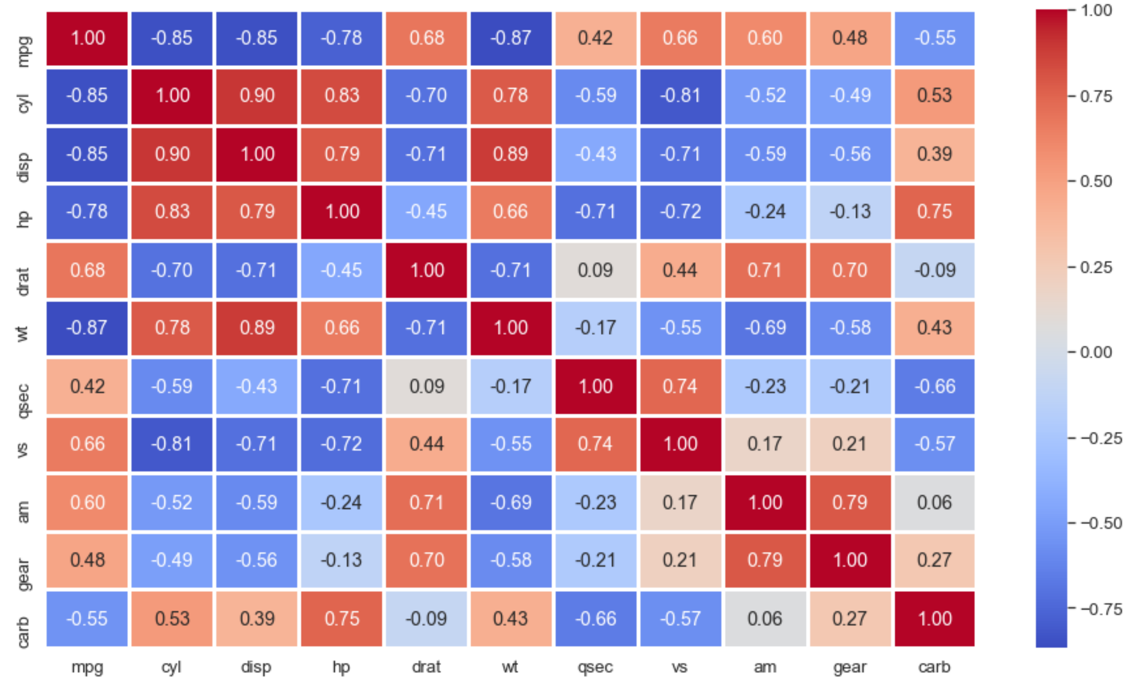
```
[13]: df.corr()
```

```
[13]:            mpg       cyl      disp        hp      drat        wt      qsec
      vs        am      gear      carb
      mpg   1.000000 -0.852162 -0.847551 -0.776168  0.681172 -0.867659  0.418684
      0.664039  0.599832  0.480285 -0.550925
      cyl  -0.852162  1.000000  0.902033  0.832447 -0.699938  0.782496 -0.591242
      -0.810812 -0.522607 -0.492687  0.526988
      disp -0.847551  0.902033  1.000000  0.790949 -0.710214  0.887980 -0.433698
      -0.710416 -0.591227 -0.555569  0.394977
```

```
hp   -0.776168  0.832447  0.790949  1.000000 -0.448759  0.658748 -0.708223
-0.723097 -0.243204 -0.125704  0.749812
drat  0.681172 -0.699938 -0.710214 -0.448759  1.000000 -0.712441  0.091205
0.440278  0.712711  0.699610 -0.090790
wt   -0.867659  0.782496  0.887980  0.658748 -0.712441  1.000000 -0.174716
-0.554916 -0.692495 -0.583287  0.427606
qsec  0.418684 -0.591242 -0.433698 -0.708223  0.091205 -0.174716  1.000000
0.744535 -0.229861 -0.212682 -0.656249
vs    0.664039 -0.810812 -0.710416 -0.723097  0.440278 -0.554916  0.744535
1.000000  0.168345  0.206023 -0.569607
am    0.599832 -0.522607 -0.591227 -0.243204  0.712711 -0.692495 -0.229861
0.168345  1.000000  0.794059  0.057534
gear  0.480285 -0.492687 -0.555569 -0.125704  0.699610 -0.583287 -0.212682
0.206023  0.794059  1.000000  0.274073
carb -0.550925  0.526988  0.394977  0.749812 -0.090790  0.427606 -0.656249
-0.569607  0.057534  0.274073  1.000000
```

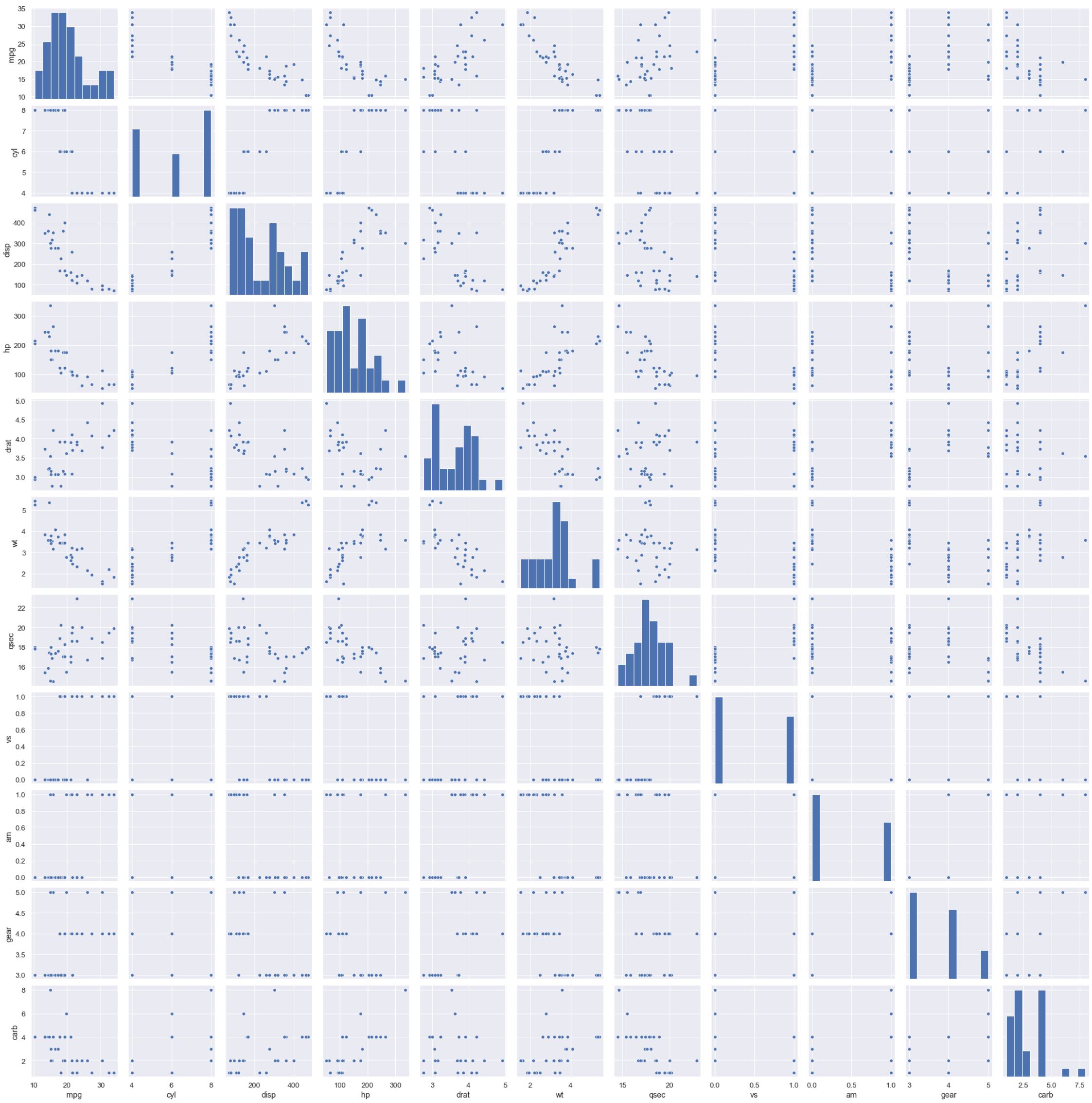
[14]:

```
plt.figure(figsize=(16,9))
sns.heatmap(df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
plt.show()
```



[15]:

```
sns.pairplot(df)
plt.show()
```



0.0.7 Data Preprocessing

0.0.8 Treat Missing Values

[16]:
```
df.isnull().sum()
```

[16]:
```
model     0
mpg       0
cyl       0
disp      0
hp        0
drat      0
wt        0
```

```
qsec     0
vs       0
am       0
gear     0
carb     0
dtype: int64
```

0.0.9 Treat Duplicate Values

[17]:

```
df.duplicated(keep='first').sum()
```

[17]:

```
0
```

0.0.10 Drop unwanted features

[18]:

```
df.columns
```

[18]:

```
Index(['model', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am',
'gear', 'carb'], dtype='object')
```

[19]:

```
df.drop(['model'],axis=1,inplace=True)
```

[20]:

```
df
```

[20]:

```
     mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb
0   21.0    6  160.0  110  3.90  2.620  16.46   0   1     4     4
1   21.0    6  160.0  110  3.90  2.875  17.02   0   1     4     4
2   22.8    4  108.0   93  3.85  2.320  18.61   1   1     4     1
3   21.4    6  258.0  110  3.08  3.215  19.44   1   0     3     1
4   18.7    8  360.0  175  3.15  3.440  17.02   0   0     3     2
5   18.1    6  225.0  105  2.76  3.460  20.22   1   0     3     1
6   14.3    8  360.0  245  3.21  3.570  15.84   0   0     3     4
7   24.4    4  146.7   62  3.69  3.190  20.00   1   0     4     2
8   22.8    4  140.8   95  3.92  3.150  22.90   1   0     4     2
9   19.2    6  167.6  123  3.92  3.440  18.30   1   0     4     4
10  17.8    6  167.6  123  3.92  3.440  18.90   1   0     4     4
11  16.4    8  275.8  180  3.07  4.070  17.40   0   0     3     3
12  17.3    8  275.8  180  3.07  3.730  17.60   0   0     3     3
13  15.2    8  275.8  180  3.07  3.780  18.00   0   0     3     3
14  10.4    8  472.0  205  2.93  5.250  17.98   0   0     3     4
15  10.4    8  460.0  215  3.00  5.424  17.82   0   0     3     4
16  14.7    8  440.0  230  3.23  5.345  17.42   0   0     3     4
17  32.4    4   78.7   66  4.08  2.200  19.47   1   1     4     1
18  30.4    4   75.7   52  4.93  1.615  18.52   1   1     4     2
19  33.9    4   71.1   65  4.22  1.835  19.90   1   1     4     1
20  21.5    4  120.1   97  3.70  2.465  20.01   1   0     3     1
```

```
21  15.5    8  318.0  150  2.76  3.520  16.87  0  0     3     2
22  15.2    8  304.0  150  3.15  3.435  17.30  0  0     3     2
23  13.3    8  350.0  245  3.73  3.840  15.41  0  0     3     4
24  19.2    8  400.0  175  3.08  3.845  17.05  0  0     3     2
25  27.3    4   79.0   66  4.08  1.935  18.90  1  1     4     1
26  26.0    4  120.3   91  4.43  2.140  16.70  0  1     5     2
27  30.4    4   95.1  113  3.77  1.513  16.90  1  1     5     2
28  15.8    8  351.0  264  4.22  3.170  14.50  0  1     5     4
29  19.7    6  145.0  175  3.62  2.770  15.50  0  1     5     6
30  15.0    8  301.0  335  3.54  3.570  14.60  0  1     5     8
31  21.4    4  121.0  109  4.11  2.780  18.60  1  1     4     2
```

0.0.11 Create and save processed dataset

```
[21]: #df.to_csv("carstrain.csv",index=False)
```

0.0.12 Model Training

```
[22]: df.columns
```

```
[22]: Index(['mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear',
      'carb'], dtype='object')
```

0.0.13 Using PyCaret

```
[23]: exp_reg = setup(data = df, target = 'mpg', session_id=0, normalize=True,␣
       ↪categorical_features=['vs', 'am', 'gear', 'carb'])
```

```
Setup Succesfully Completed.

<pandas.io.formats.style.Styler at 0x1c0ef41b400>
```

```
[24]: compare_models()
```

```
<pandas.io.formats.style.Styler at 0x1c0ec264640>
```

```
[24]: BayesianRidge(alpha_1=1e-06, alpha_2=1e-06, alpha_init=None,
                    compute_score=False, copy_X=True, fit_intercept=True,
                    lambda_1=1e-06, lambda_2=1e-06, lambda_init=None, n_iter=300,
                    normalize=False, tol=0.001, verbose=False)
```