importing liabraries

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib as plt
import matplotlib.pyplot as plt
```

```
cars = pd.read_csv("/content/auto_mpg.csv", names=None)
```

```
names = ['MPG','cylinders','Displacement', 'Horsepower', 'weight','Acceleration','Model Year','Origin','car Name']
```

```
cars.columns= names
```

```
cars.head()
```

	MPG	cylinders	Displacement	Horsepower	weight	Acceleration	Model Year	Origin	car Name
0	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
1	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
2	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
3	17.0	8	302.0	140	3449	10.5	70	1	ford torino
4	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500

Saved successfully!

```
(397, 9)
```

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   MPG           397 non-null    float64
 1   cylinders     397 non-null    int64
 2   Displacement  397 non-null    float64
 3   Horsepower    397 non-null    object
 4   weight        397 non-null    int64
 5   Acceleration  397 non-null    float64
 6   Model Year    397 non-null    int64
 7   Origin        397 non-null    int64
 8   car Name      397 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
```

```
cars["Horsepower"].isnull().sum()
```

```
0
```

```
cars=cars[cars["Horsepower"] != '?']
print(cars.shape)
```

```
(391, 9)
```

```
cars["Horsepower"]=cars["Horsepower"].astype('float')
print(cars.dtypes)
```

```
MPG             float64
cylinders         int64
Displacement    float64
Horsepower      float64
weight            int64
```

```
Acceleration    float64
Model Year        int64
Origin            int64
car Name         object
dtype: object
<ipython-input-10-05d0b84fe0b6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  cars["Horsepower"]=cars["Horsepower"].astype('float')
```

```
cars.isnull().sum()
```

```
MPG             0
cylinders       0
Displacement    0
Horsepower      0
weight          0
Acceleration    0
Model Year      0
Origin          0
car Name        0
dtype: int64
```

Exploratory Analysis

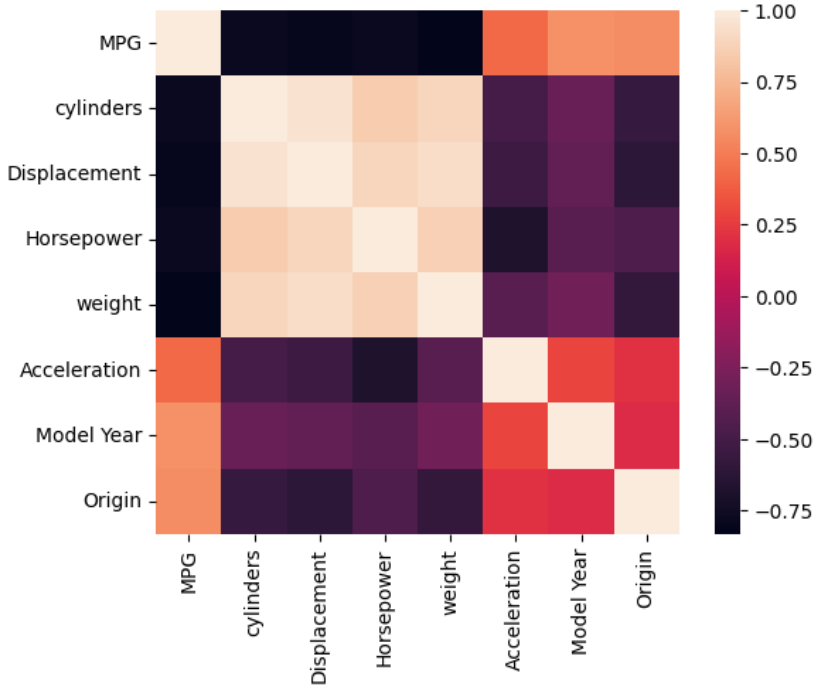
```
cars["MPG"].describe()
```

```
count    391.000000
mean      23.459847
std        7.810128
min        9.000000
25%       17.000000
max       46.600000
Name: MPG, dtype: float64
```

Saved successfully!

```
import seaborn as sns
corrmatrix = cars.corr()
sns.heatmap(corrmatrix, square= True)
```

```
<ipython-input-13-c470deefec00>:2: FutureWarning: The default value of numeric_only in DataFrame.corr i
  corrmatrix = cars.corr()
<Axes: >
```



```
from sklearn.metrics import mean_squared_error

features = ['cylinders','Displacement', 'Horsepower', 'weight','Acceleration','Model Year','Origin']
X = cars[features]
y = cars["MPG"]

X_train, X_test, y_train, y_test= train_test_split(X ,y, test_size=0.3,random_state=24)


regr =LinearRegression()
regr.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
predict_values= regr.predict(X_test)


predict_values
```

```
array([28.98319224, 26.39187916, 30.73892334, 17.54753575, 20.58585136,
       35.14751626, 29.07981513,  7.64073996, 25.54412284, 30.66116118,
       25.76382205, 10.311702  , 28.15407674, 20.68238756,  8.74694083,
       31.63534947, 31.16491326, 23.00018623, 24.09915332, 13.69535809,
       22.32753461, 24.58963406, 21.34618359, 28.64121716, 26.99022811,
       21.38332404, 26.62313065, 28.89539988, 36.13286973, 19.09332342,
       28.04784449, 30.59681937, 26.63209749, 11.65172323, 25.34278108,
       32.13890595, 31.32534439, 26.77727555, 12.32898894, 22.44821988,
       31.4342112 , 21.62424754, 26.05234347, 23.59296996, 28.74290616,
       17.07365939, 24.51134573, 23.47447188, 32.20856975, 25.69998714,
       19.00245374, 14.95690784, 26.39286337, 17.16613262, 25.62358536,
       30.31047022, 11.71617086, 32.66585635, 36.00997986, 26.58037419,
       19.2861861 , 24.19554281, 13.34353237, 23.06212512, 22.24769022,
       15.47359274, 22.72377863, 11.40965781, 29.13246907, 26.61513232,
       32.72317287, 24.88366239, 14.66364686, 29.98982358, 10.42772646,
       23.26801035, 16.52697645, 20.56773661, 22.08637274, 30.10965011,
       20.13948653, 29.05896207, 33.52760008, 30.08491344, 30.50880092,
       [illegible]  [illegible]  24.59317393, 29.78036465, 27.96276329,
       [illegible]  [illegible]  15.45911506, 31.01542712, 14.62852497,
       27.79287127, 17.27065529, 21.35730963, 27.80303003,  9.74876576,
       16.0608083 , 22.22404819,  6.45204073, 10.69077454, 23.85933183,
       29.5020639 , 22.71141174, 26.47561322, 33.46850582, 17.17617848,
       26.07545448, 32.27357293, 26.03072362, 26.91658339, 12.03018816,
       25.15404052, 10.0122611 , 20.95478654])
```

Saved successfully!

```
pd.DataFrame(np.c_[X_test,y_test,predict_values],columns = ['cylinders','Displacement', 'Horsepower', 'weight','Acceleration','M
```

	cylinders	Displacement	Horsepower	weight	Acceleration	Model Year	Origin	MPG	mpg new
0	4.0	119.0	82.0	2720.0	19.4	82.0	1.0	31.0	28.983192
1	4.0	107.0	86.0	2464.0	15.5	76.0	2.0	28.0	26.391879
2	4.0	98.0	80.0	1915.0	14.4	79.0	1.0	35.7	30.738923
3	8.0	302.0	130.0	3870.0	15.0	76.0	1.0	13.0	17.547536
4	4.0	110.0	87.0	2672.0	17.5	70.0	2.0	25.0	20.585851
...	...	...	...	...	...	...	...	...	...
113	4.0	122.0	96.0	2300.0	15.5	77.0	1.0	25.5	26.916583
114	8.0	351.0	158.0	4363.0	13.0	73.0	1.0	13.0	12.030188
115	4.0	91.0	70.0	1955.0	20.5	71.0	1.0	26.0	25.154041
116	8.0	440.0	215.0	4312.0	8.5	70.0	1.0	14.0	10.012261
117	4.0	121.0	112.0	2868.0	15.5	73.0	2.0	19.0	20.954787

118 rows × 9 columns

```
from sklearn.metrics import r2_score
print(r2_score(y_test,predict_values))
```

```
0.8083852060985331
```

```
regr.score(X_test, y_test)
```

```
0.8083852060985331
```

RandomForest

```
from sklearn.ensemble import RandomForestRegressor

forest_reg =RandomForestRegressor(random_state=12)
forest_reg.fit(X_train, y_train)
print("Training accuracy:",forest_reg.score(X_train,y_train))

y_pred= forest_reg.predict(X_test)
print("Testing accuracy:",forest_reg.score(X_test, y_test))
```

```
Training accuracy: 0.9818807392348246
Testing accuracy: 0.8943718899779411
```

```
MSE = mean_squared_error(y_test, predict_values)
print(MSE)
```

```
10.13592677140194
```

Deployment

```
# save the Model
import joblib
joblib.dump(forest_reg,"Auto_MPG_predictor_model.pkl")
```

Saved successfully!

```
model= joblib.load("Auto_MPG_predictor_model.pkl")
```

```
model.predict(X_test)
```

```
array([27.916, 24.029, 32.798, 15.397, 23.081, 36.944, 35.688, 12.9  ,
       24.441, 28.135, 24.997, 13.395, 28.392, 20.019, 12.43 , 33.972,
       31.901, 24.246, 24.068, 14.375, 22.926, 25.052, 18.855, 29.479,
       25.967, 21.073, 27.225, 29.68 , 36.096, 18.663, 27.642, 35.618,
       24.609, 13.037, 22.63 , 35.748, 31.658, 29.175, 13.74 , 20.898,
       33.523, 17.63 , 22.778, 27.39 , 27.603, 18.022, 22.246, 22.519,
       31.336, 24.064, 18.921, 14.715, 24.699, 17.478, 27.787, 29.119,
       13.385, 33.3  , 36.399, 26.28 , 21.931, 20.942, 14.445, 22.335,
       21.263, 16.548, 19.582, 14.677, 27.61 , 23.728, 37.068, 26.125,
       14.33 , 30.995, 12.895, 19.552, 15.657, 18.308, 21.983, 31.454,
       17.91 , 28.662, 34.384, 31.701, 30.803, 32.855, 21.916, 21.778,
       24.23 , 30.21 , 18.695, 17.895, 14.27 , 32.024, 16.347, 29.998,
       16.648, 19.68 , 26.58 , 13.115, 18.031, 20.318, 12.75 , 14.122,
       24.11 , 31.629, 24.766, 25.259, 36.298, 15.903, 23.255, 33.745,
       24.041, 24.937, 13.915, 29.316, 14.55 , 21.683])
```

launch & monitor

```
regr.save('/content/drive')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-26-6059dd96a157> in <cell line: 1>()
----> 1 regr.save('/content/drive')

AttributeError: 'LinearRegression' object has no attribute 'save'
```

SEARCH STACK OVERFLOW

```
#path = '/content/drive//content/Auto_MPG_predictor_model.pkl'

model.save(path)
```

DEPLOYMENT

```
??
```

```
?
```

Saved successfully!

0s completed at 7:24 PM