Name Haoyue Chang
Batch code< LISUM22>
Submission date: June 27th, 2023
Submitted to Deployment on Flask for Data Glacier Internship

In [23]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
# For randomized data splitting
from sklearn.model_selection import train_test_split

# To build linear regression_model
import statsmodels.api as sm
# To check model performance
from sklearn.metrics import mean_absolute_error, mean_squared_error
import pickle
```

In [24]:
```
df = pd.read_csv('/Users/haoyuechang/Desktop/auto-mpg.csv')
#cData = pd.read_csv("auto-mpg.csv")
```

:
```
df = pd.read_csv('/Users/haoyuechang/Desktop/auto-mpg.csv')
#cData = pd.read_csv("auto-mpg.csv")
```

:
```
#Data processing
df.info()
#398 entries, 9 columns
```

```
<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    object 
 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(3), int64(4), object(2)
memory usage: 28.1+ KB
```

:
```
#drop car name since it is not useful to analyze
df1 = df.drop(["car name"], axis=1)
```

```
hpIsDigit = pd.DataFrame(
    df1.horsepower.str.isdigit()
)  # if the string is made of digits store True else False

# print the entries where isdigit = False
df1[hpIsDigit["horsepower"] == False]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
32	25.0	4	98.0	?	2046	19.0	71	1
126	21.0	6	200.0	?	2875	17.0	74	1
330	40.9	4	85.0	?	1835	17.3	80	2
336	23.6	4	140.0	?	2905	14.3	80	1
354	34.5	4	100.0	?	2320	15.8	81	2
374	23.0	4	151.0	?	3035	20.5	82	1

```
df1= df1.replace("?", np.nan)
df1[hpIsDigit["horsepower"] == False]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
32	25.0	4	98.0	NaN	2046	19.0	71	1
126	21.0	6	200.0	NaN	2875	17.0	74	1
330	40.9	4	85.0	NaN	1835	17.3	80	2
336	23.6	4	140.0	NaN	2905	14.3	80	1
354	34.5	4	100.0	NaN	2320	15.8	81	2
374	23.0	4	151.0	NaN	3035	20.5	82	1

]:
```
df1.median()
```

]:
```
mpg              23.0
cylinders         4.0
displacement    148.5
horsepower       93.5
weight         2803.5
acceleration     15.5
model year       76.0
origin            1.0
dtype: float64
```

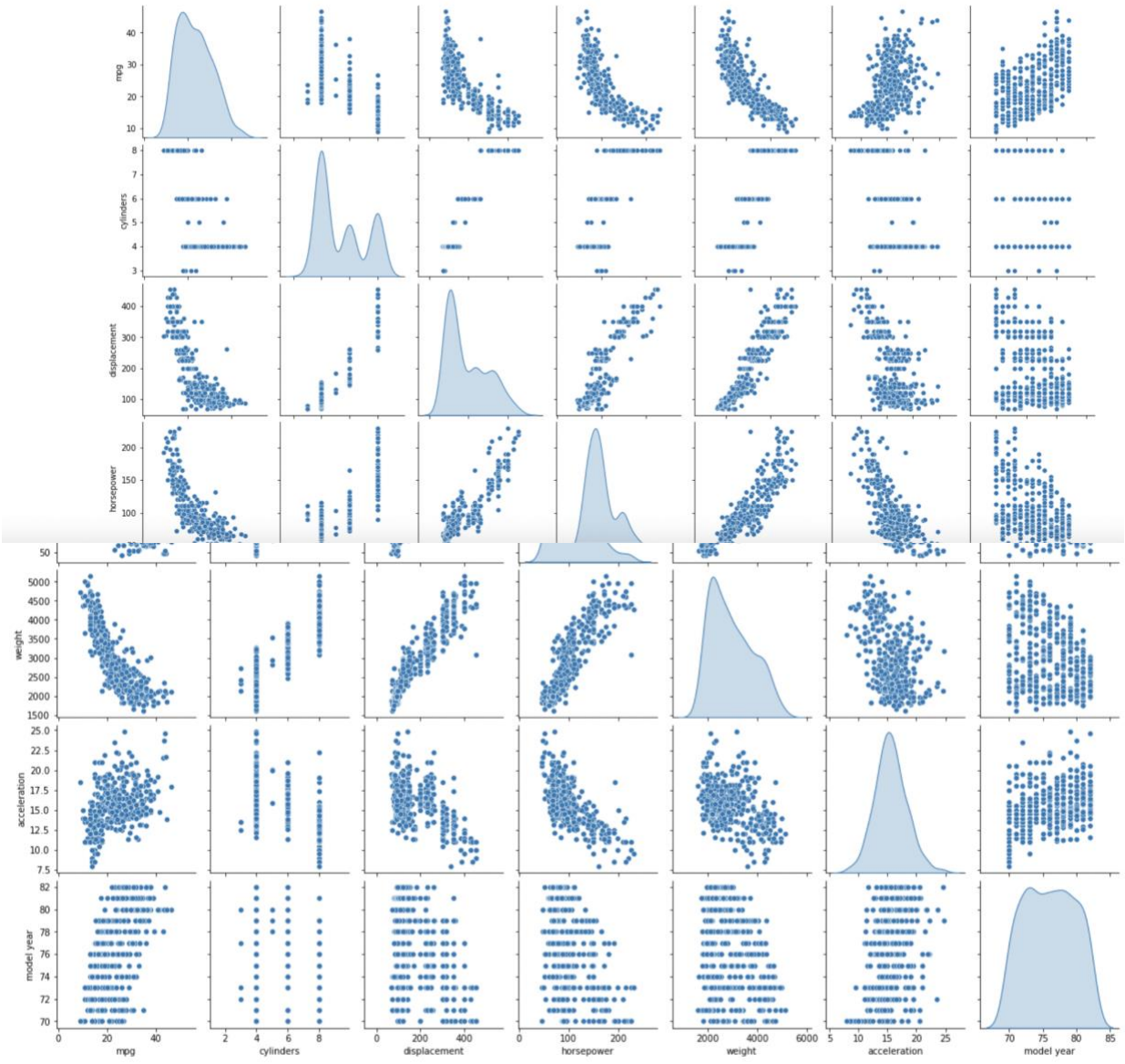
]:
```
# Let's replace the missing values with median values of the columns.
# Note that we do not need to specify the column names below.
# Every column's missing value is replaced with that column's median respectively

medianFiller = lambda x: x.fillna(x.median())
df1 = df1.apply(medianFiller, axis=0)
```

]:
```
# let's convert the horsepower column from object type to float type
df1["horsepower"] = df1["horsepower"].astype(float)
```

In [32]:
```
#Bivariate
df_attr = df1.iloc[:, 0:7]
sns.pairplot(
    df_attr, diag_kind="kde")
```

Out[32]: <seaborn.axisgrid.PairGrid at 0x7fb13015ca00>



In [33]:
```
# drop_first=True will drop one of the three origin columns
#vreat dummy variables
df2 = pd.get_dummies(df1, columns=["origin"], drop_first=True)
df2.head()
```

Out[33]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin_2	origin_3
0	18.0	8	307.0	130.0	3504	12.0	70	0	0
1	15.0	8	350.0	165.0	3693	11.5	70	0	0
2	18.0	8	318.0	150.0	3436	11.0	70	0	0
3	16.0	8	304.0	150.0	3433	12.0	70	0	0
4	17.0	8	302.0	140.0	3449	10.5	70	0	0

In [34]:
```
#SPLIT DATA
# independent variables
X = df2.drop(["mpg"], axis=1)
# dependent variable
y = df2[["mpg"]]
```

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=1)
```

```
print(X_train.head())
```

```
     cylinders  displacement  horsepower  weight  acceleration  model year  \
350          4         105.0        63.0    2215          14.9          81
59           4          97.0        54.0    2254          23.5          72
120          4         121.0       112.0    2868          15.5          73
12           8         400.0       150.0    3761           9.5          70
349          4          91.0        68.0    1985          16.0          81

     origin_2  origin_3
350         0         0
59          1         0
120         1         0
12          0         0
349         0         1
```

```
print(X_test.head())
```

```
     cylinders  displacement  horsepower  weight  acceleration  model year  \
174          6         171.0        97.0    2984          14.5          75
359          4         141.0        80.0    3230          20.4          81
250          8         318.0       140.0    3735          13.2          78
274          5         131.0       103.0    2830          15.9          78
283          6         232.0        90.0    3265          18.2          79

     origin_2  origin_3
174         0         0
359         1         0
250         0         0
274         1         0
283         0         0
```

```
olsmod = sm.OLS(y_train, X_train)
olsres = olsmod.fit()
```

```
print(olsres.summary())
```

```
                                 OLS Regression Results
================================================================================================
Dep. Variable:                      mpg   R-squared (uncentered):                   0.980
Model:                              OLS   Adj. R-squared (uncentered):              0.980
Method:                   Least Squares   F-statistic:                              1689.
Date:                  Mon, 26 Jun 2023   Prob (F-statistic):                   1.04e-225
Time:                          16:45:30   Log-Likelihood:                         -741.29
No. Observations:                   278   AIC:                                      1499.
Df Residuals:                       270   BIC:                                      1528.
Df Model:                             8
Covariance Type:              nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
cylinders       -0.6654      0.427     -1.558      0.120      -1.506       0.175
displacement     0.0284      0.010      2.747      0.006       0.008       0.049
horsepower      -0.0443      0.016     -2.845      0.005      -0.075      -0.014
weight          -0.0067      0.001     -7.945      0.000      -0.008      -0.005
acceleration    -0.1202      0.110     -1.090      0.277      -0.337       0.097
model year       0.6229      0.029     21.627      0.000       0.566       0.680
origin_2         2.3585      0.699      3.372      0.001       0.982       3.735
origin_3         2.1483      0.697      3.083      0.002       0.777       3.520
==============================================================================
Omnibus:                       27.282   Durbin-Watson:                   2.241
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               42.908
Skew:                           0.609   Prob(JB):                     4.82e-10
Kurtosis:                       4.490   Cond. No.                     1.24e+04
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.24e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
# Saving model to disk
pickle.dump(olsres, open('model.pkl','wb'))
```

```
# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[8,307.0,130.0,3504,12.0,70,0,0]]))
```

```
[16.27077424]
```

```
!pip3 install flask==2.2.2
```

```
Collecting flask==2.2.2
  Using cached Flask-2.2.2-py3-none-any.whl (101 kB)
Requirement already satisfied: Werkzeug>=2.2.2 in /opt/anaconda3/lib/python3.9/site-packages (from flask==2.2.2) (2.
3.6)
Requirement already satisfied: Jinja2>=3.0 in /opt/anaconda3/lib/python3.9/site-packages (from flask==2.2.2) (3.1.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /opt/anaconda3/lib/python3.9/site-packages (from flask==
2.2.2) (4.8.1)
Requirement already satisfied: itsdangerous>=2.0 in /opt/anaconda3/lib/python3.9/site-packages (from flask==2.2.2)
(2.1.2)
Requirement already satisfied: click>=8.0 in /opt/anaconda3/lib/python3.9/site-packages (from flask==2.2.2) (8.1.3)
Requirement already satisfied: zipp>=0.5 in /opt/anaconda3/lib/python3.9/site-packages (from importlib-metadata>=3.6.
0->flask==2.2.2) (3.6.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/anaconda3/lib/python3.9/site-packages (from Jinja2>=3.0->flask
==2.2.2) (2.1.2)
Installing collected packages: flask
  Attempting uninstall: flask
    Found existing installation: Flask 2.3.2
    Uninstalling Flask-2.3.2:
      Successfully uninstalled Flask-2.3.2
Successfully installed flask-2.2.2
```

```python
import numpy as np
from flask import Flask, request,render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='a car mileage would be  {}'.format(output))

if __name__ == "__main__":
    app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
```

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instea
d.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [26/Jun/2023 17:50:08] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [26/Jun/2023 17:50:08] "GET /static/css/style.css HTTP/1.1" 404 -
```

Predict MVG

Cylinders	Displacement	Horsepower	weight	Acceleration	Model year	Origin_2	Origin_3	Predict

a car mileage would be 17.47