Python Lab1 Assignment DATA 622 Spring 2026

Umais Siddiqui

Invalid Date

Table of contents

Introduction:

The purpose of this assignment is to review basic data manipulation and plotting in python.

1 Problem 1

1.1 Inspection of the data

In the code below I am loading the required libraries to load the file that is needed to answer Problem 1. The data I will be loading is the Auto data. Finally making that the missing values have been removed from the data.

```
mpg              float64
cylinders          int64
displacement     float64
horsepower       float64
weight             int64
acceleration     float64
year               int64
origin             int64
name                 str
dtype: object
```

1.2 (A) Which of the predictors are quantitative, and which are qualitative?

Quantitative: mpg, cylinders, displacement, horsepower, weight, acceleration, year.

Qualitative: origin, name.

```
min    1
max    3
Name: origin, dtype: int64
```

In this dataset, “origin” is usually coded as 1, 2, or 3 (representing American, European, and Japanese cars respectively). Unlike cylinders, where an 8-cylinder engine actually has twice the physical components of a 4-cylinder engine, the numbers for origin carry no such weight

1.3 (B) What is the range of each quantitative predictor? You can answer this using the min() and max() methods in numpy.

```
      mpg  cylinders  displacement  horsepower  weight  acceleration  year
min   9.0          3          68.0        46.0    1613           8.0    70
max  46.6          8         455.0       230.0    5140          24.8    82
```

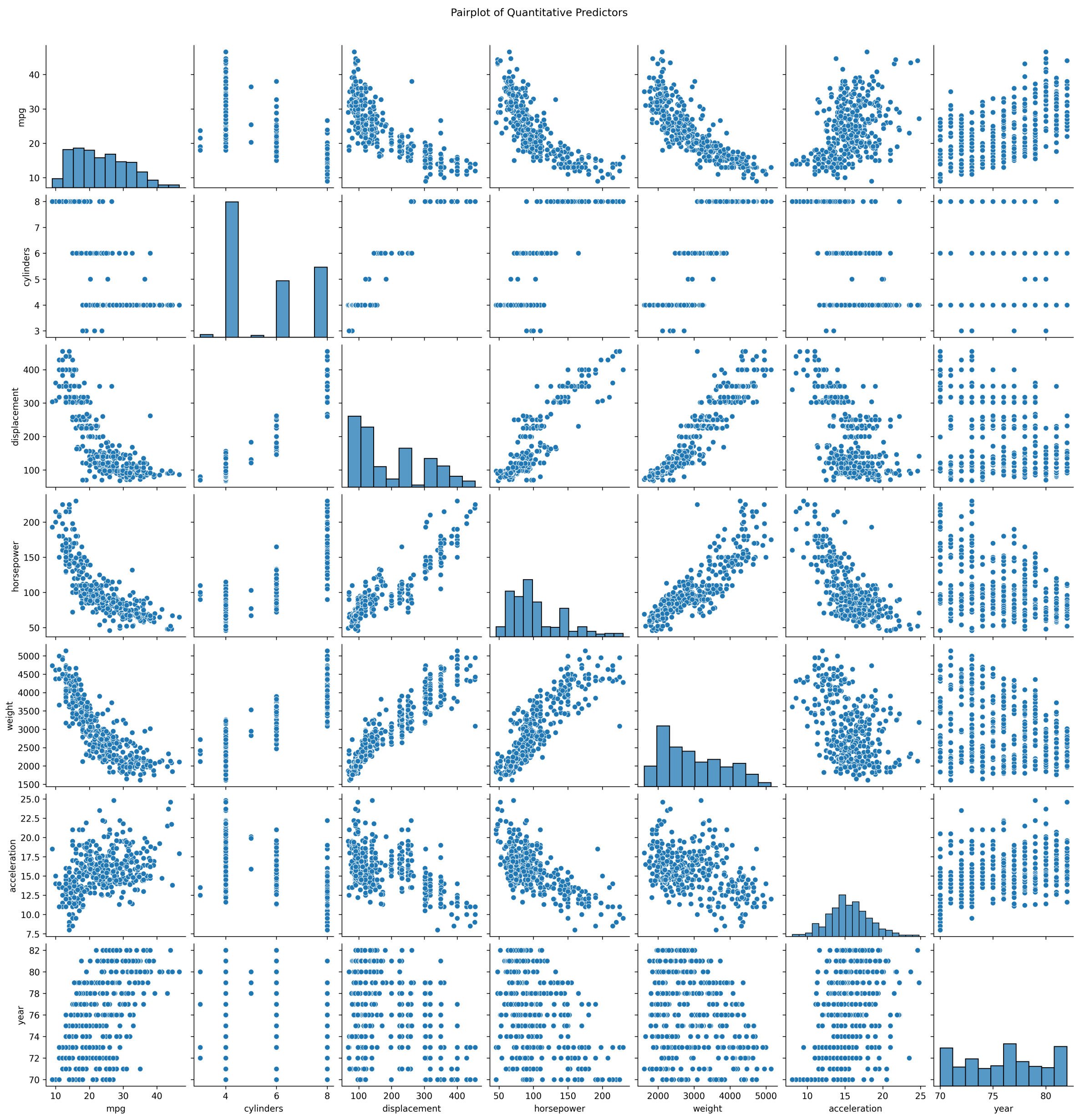
1.4 (C) What is the mean and standard deviation of each quantitative predictor?

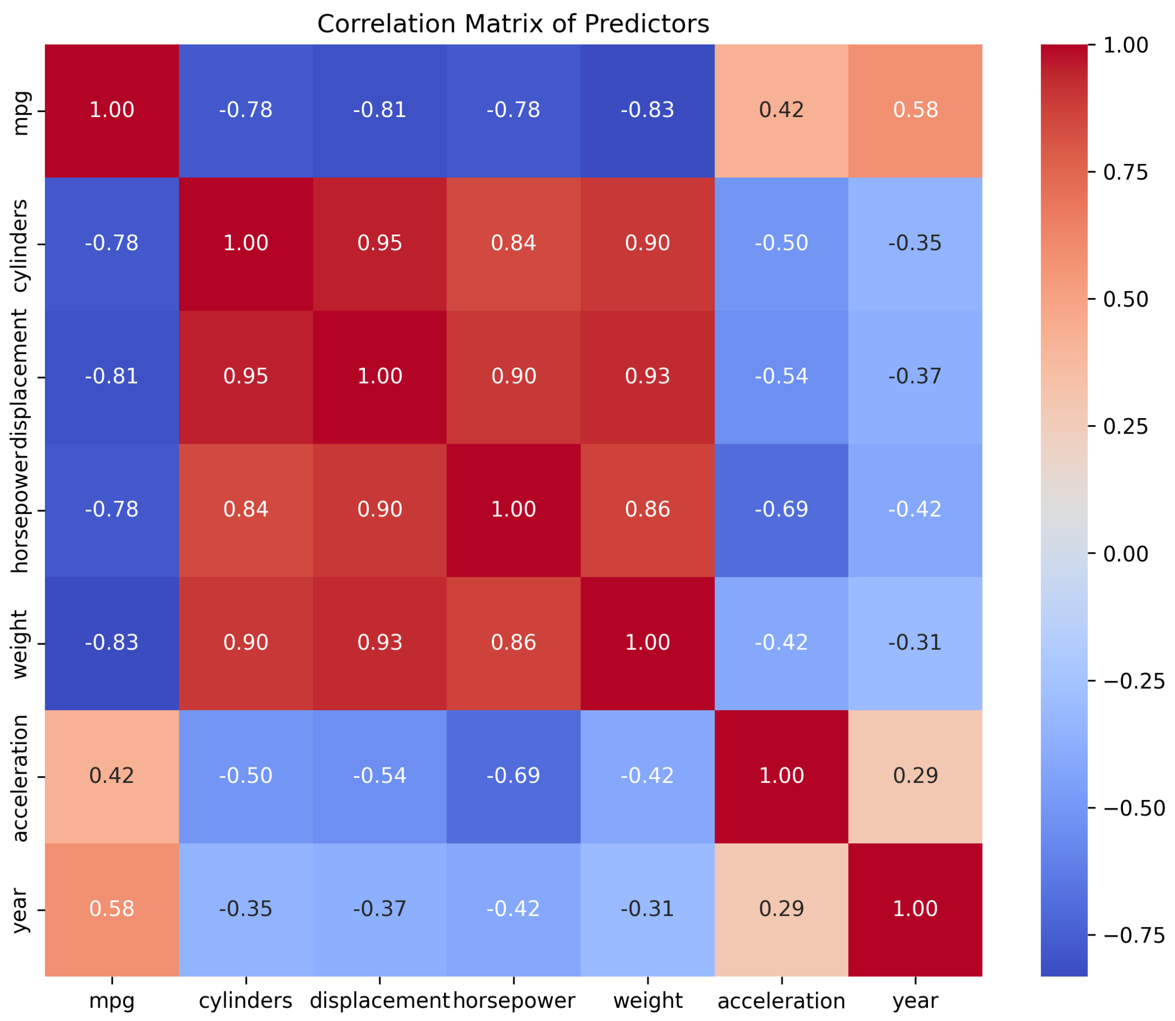
```
            mpg  cylinders  displacement  horsepower       weight  \
mean  23.445918   5.471939    194.411990  104.469388  2977.584184
std    7.805007   1.705783    104.644004   38.491160   849.402560

      acceleration       year
mean     15.541327  75.979592
std       2.758864   3.683737
```

1.5 (D) Now remove the 10th through 85th observations. What is the range, mean, and standard deviation of each predictor in the subset of the data that remains?

```
      mpg  cylinders  displacement  horsepower  weight  acceleration  year
min  11.0          3          68.0        46.0    1649           8.5    70
max  46.6          8         455.0       230.0    4997          24.8    82
            mpg  cylinders  displacement  horsepower       weight  \
mean  24.404430   5.373418    187.240506  100.721519  2935.971519
std    7.867283   1.654179     99.678367   35.708853   811.300208

      acceleration       year
mean     15.726899  77.145570
std       2.693721   3.106217
```

1.6 (E) Using the full data set, investigate the predictors graphically, using scatterplots or other tools of your choice. Create some plots highlighting the relationships among the predictors. Comment on your findings.





Strong Negative Correlations (Predicting mpg):

Based on the scatterplots and correlation matrix, the following predictors show strong negative correlations with mpg:

Weight vs. MPG: - There is a very strong negative linear relationship. - As weight increases, miles per gallon decreases significantly. - Weight is one of the most powerful predictors. - We can see that the correlation coefficient is around -0.83.

Displacement/Horsepower vs. MPG:

- Similar to weight, as engine size and power increase, fuel efficiency drops.
- These relationships appear slightly non-linear (curved), suggesting that a simple linear regression might need polynomial terms for a perfect fit.
- The correlation coefficients are around -0.81 for displacement and -0.78 for horsepower.

Strong Positive Correlations (Collinearity):

Displacement, Horsepower, and Weight:

- These three variables are highly positively correlated with each other > 0.86.
- This indicates that larger engines tend to be heavier and produce more power.

Machine Learning Impact: In a regression model, this is called Multicollinearity. Including all three might be redundant.

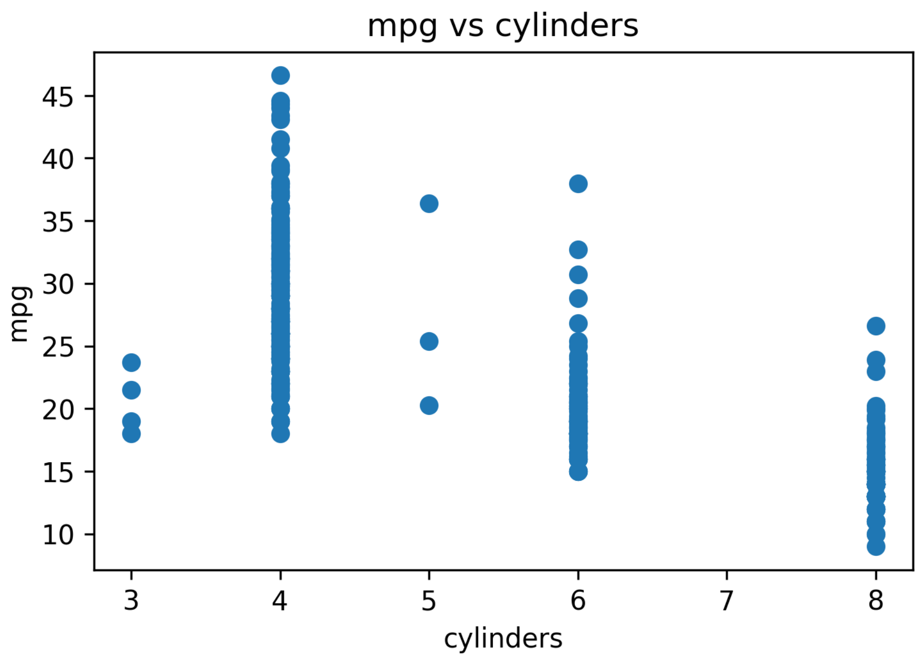
The Role of 'Year' and 'Cylinders'

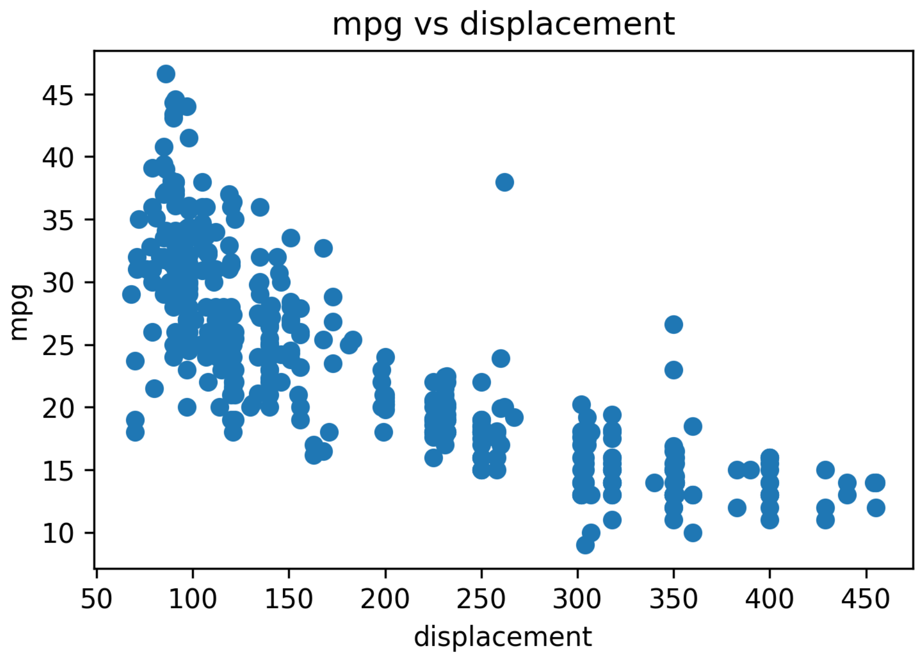
Year vs. MPG: I am seeing a general upward trend. Newer cars tend to have better fuel efficiency, likely due to advancements in technology and stricter emissions standards.

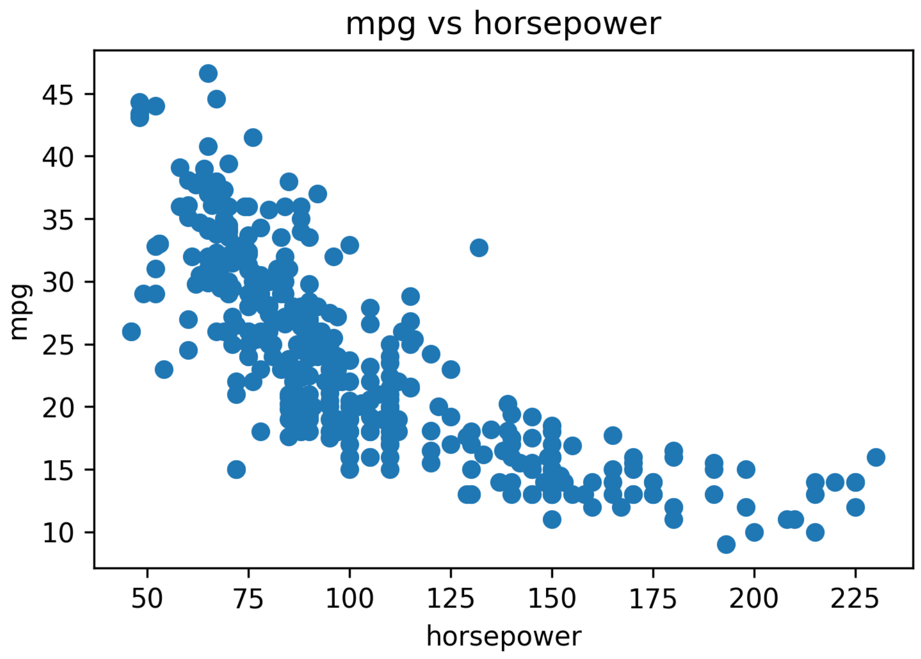
Cylinders: Although discrete, the scatterplots show distinct "strips." Each strip corresponds to a different number of cylinders, with higher cylinder counts generally associated with lower mpg. Notice that cylinders have a moderate negative correlation with mpg (-0.78).

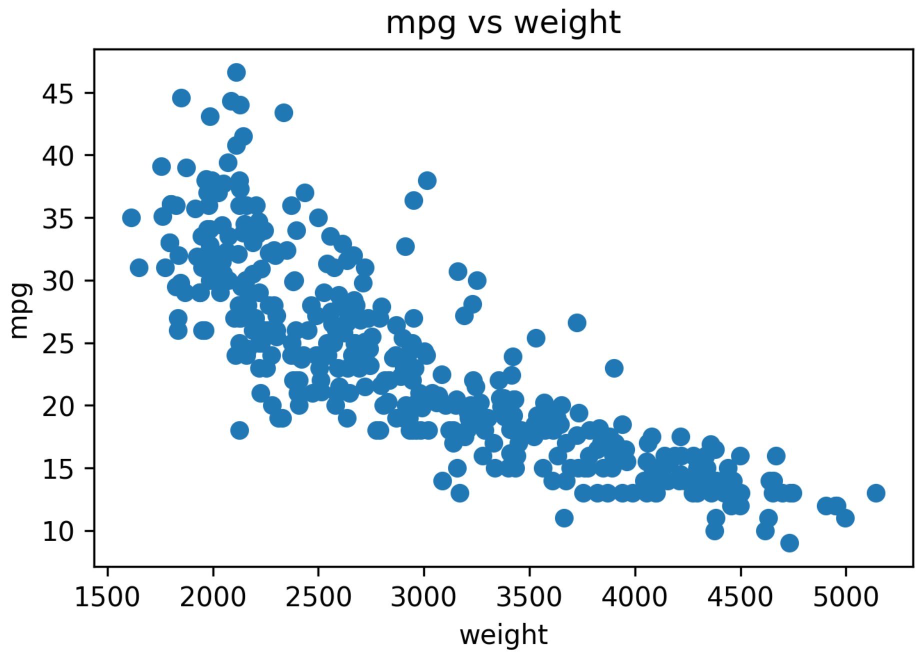
Potential Outliers: There are a few points that deviate significantly from the general trends, particularly in horsepower, weight and acceleration. These could be outliers or special cases (e.g., sports cars).

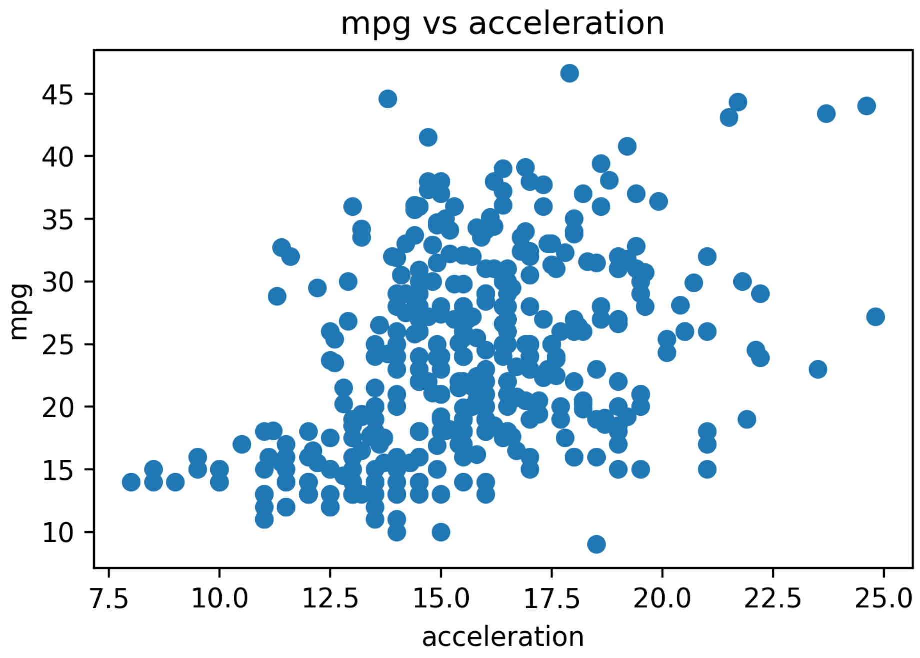
1.7 (F) Suppose that we wish to predict gas mileage (mpg) on the basis of the other variables. Do your plots suggest that any of the other variables might be useful in predicting mpg? Justify your answer.

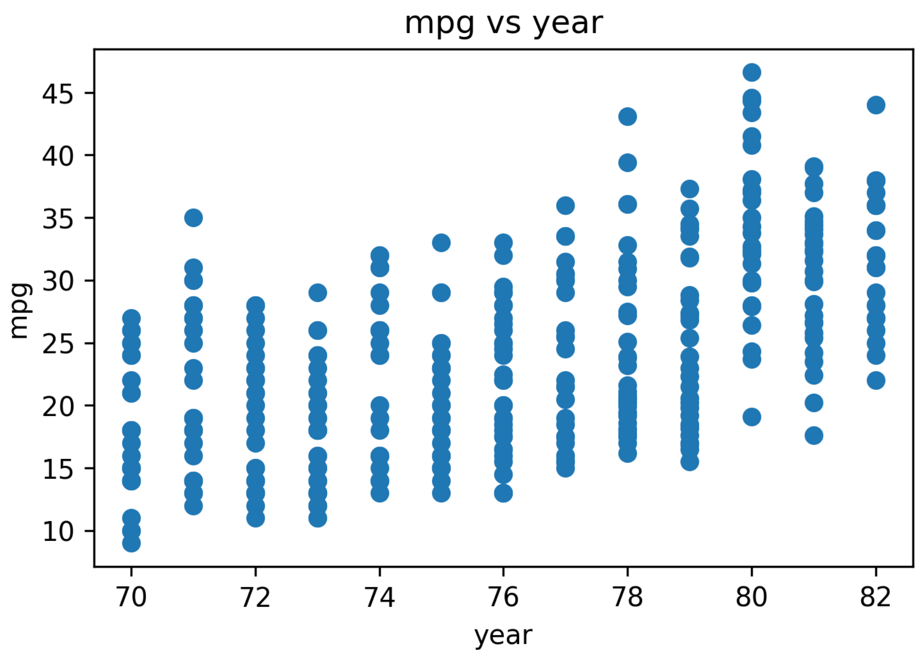





mpg vs horsepower
mpg
45
40
35
30
25
20
15
10
50
75
100
125
150
175
200
225
horsepower


mpg vs weight
mpg
45
40
35
30
25
20
15
10
1500
2000
2500
3000
3500
4000
4500
5000
weight





Based on the scatter plots generated, several variables appear to be strong predictors for mpg:

Weight, Displacement, and Horsepower: These show strong negative relationships with mpg. As these values increase, mpg consistently drops, indicating they are primary predictors.

Year: There is a clear positive trend, showing that newer cars generally achieve better gas mileage.

Cylinders: Although discrete, the clear grouping shows that cars with more cylinders are restricted to the lower mpg range.

below is a correlation summary for mpg with other predictors:

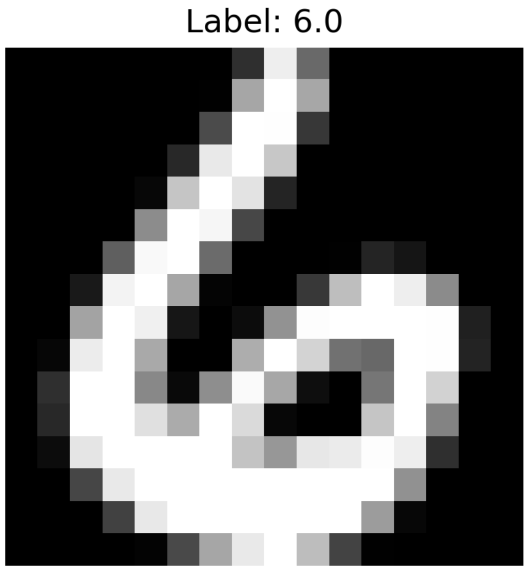
```
mpg             1.000000
cylinders      -0.777618
displacement   -0.805127
horsepower     -0.778427
weight         -0.832244
acceleration    0.423329
year            0.580541
Name: mpg, dtype: float64
```

In conclusion, the visual evidence suggests that a combination of engine size (displacement/cylinders), power (horsepower), and mass (weight) will be highly effective in predicting a vehicle's fuel efficiency.

2 Problem 2 kNN Classification of the ZIP code digit data

2.1 In order to complete this problem you will need to download zip.train and zip.test from the course website. These datasets contain images of hand-drawn digits. We will be experimenting with kNN classification and factors impacting the bias-variance trade-off, and this will also be chance to practice using scikit-learn.

2.2 a. Download and load the training and test data sets using pandas. Make sure to load all of the data (there is no header) The zeroth column corresponds to the class label, a digit from 0-9, and the columns 1 to 256 correspond to a grayscale value from -1 to 1. Select the first entry in the training set, resize it to 16x16, and plot the image (you can use plt.imshow()).



2.3 b. The following code fits imports the kNN classification function from scikit-learn as well as an accuracy function, trains a classifier, and tests its accuracy on hypothetical training and testing data:

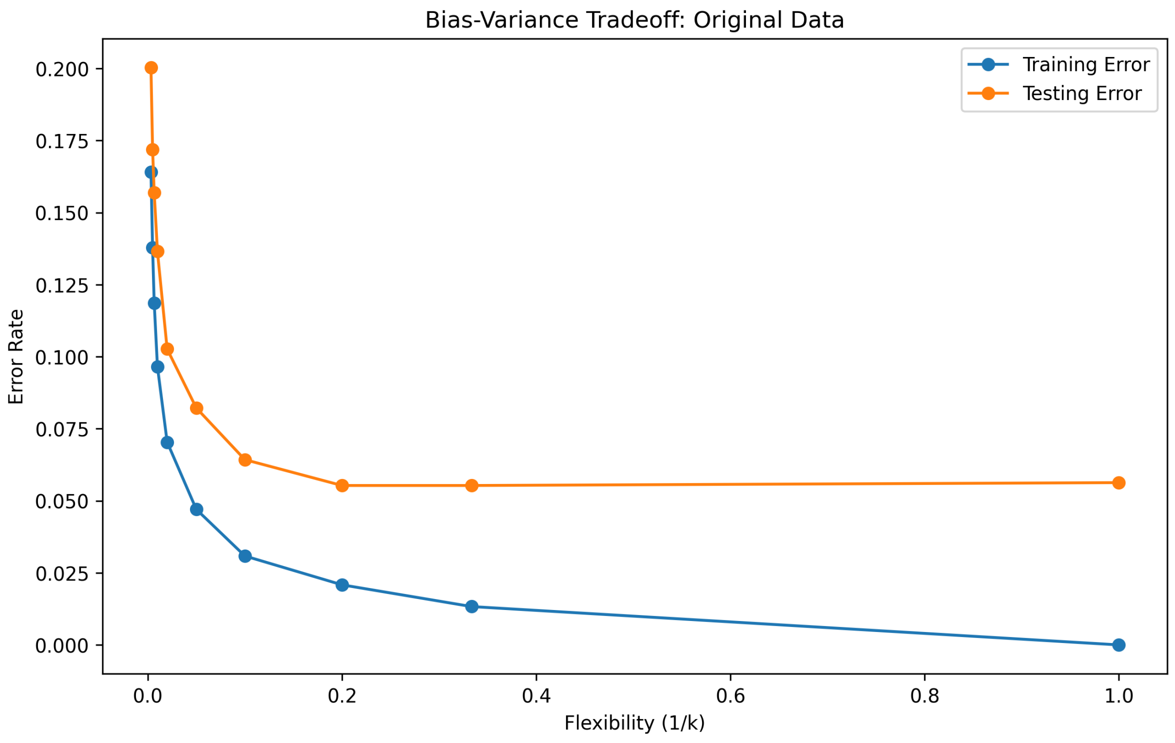
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier(n_neighbors=4)
```

```
knn.fit(zip_image_train, zip_class_train)

zip_pred_train = knn.predict(zip_image_train)
accuracy = accuracy_score(zip_class_train, zip_pred_train)
print(f'Accuracy: {accuracy:.4f}')

zip_pred_test = knn.predict(zip_image_test)
accuracy = accuracy_score(zip_class_test, zip_pred_test)
print(f'Accuracy: {accuracy:.4f}')
```

```
Accuracy: 0.9431988041853513
```

Where the dataframes contain just the training and testing images and class.Adapt this code to determine if the we can observe the bias variance trade-off for different numbers of neighbors . Specifically, recreate plot 2.17 (pay attention to the x-axis scale and use the same choice as in the book) which shows test and train classification accuracy as a function of . Select a range of from 1 to 300. You do not have to plot every single value in this range if the problem is computationally intensive on your machine.

Do you observe a shaped curve in the testing error (and a divergence from training error) as increases?



The Overfitting Zone (Right Side, $1/k = 1.0$):At the far right, $k = 1$.

We can notice that Training Error is exactly 0.000. This is because the model has "memorized" the training set. However, the Testing Error is higher than its minimum point. This gap represents High Variance.The Underfitting Zone (Left Side, $1/k \approx 0.0$):As you move to the left (where $k = 300$), both the Training and Testing errors climb significantly. The model is now too simple to capture the patterns in the digits. This represents High Bias.The Optimal Flexibility:The "Sweet Spot" is the lowest point on the orange line (the testing error). On the graph, this appears to be around $1/k = 0.2$ (which is $k = 5$). This is where the model generalizes best to new data.

Yes, a distinct U-shaped curve is observed in the testing error as flexibility ($1/k$) increases. While the training error decreases monotonically toward zero, the testing error hits a minimum at approximately $k = 5$ and then begins to increase. This divergence at high flexibility levels ($k = 1$ or 3) is a clear indication of overfitting, where the model captures local noise in the training set that does not generalize to the test set.

2.4 c.

Introduce some noise in the training and testing labels for both the training and testing data. You can do this by using np.random.choice to sample from the range of indices of each of the training and test set to determine which labels will be changed, and np.random.choice again to pick the new label. After making this modification, repeat problem (b). How did adding label noise impact the shape of the testing and training error versus curves?

Introducing Label Noise

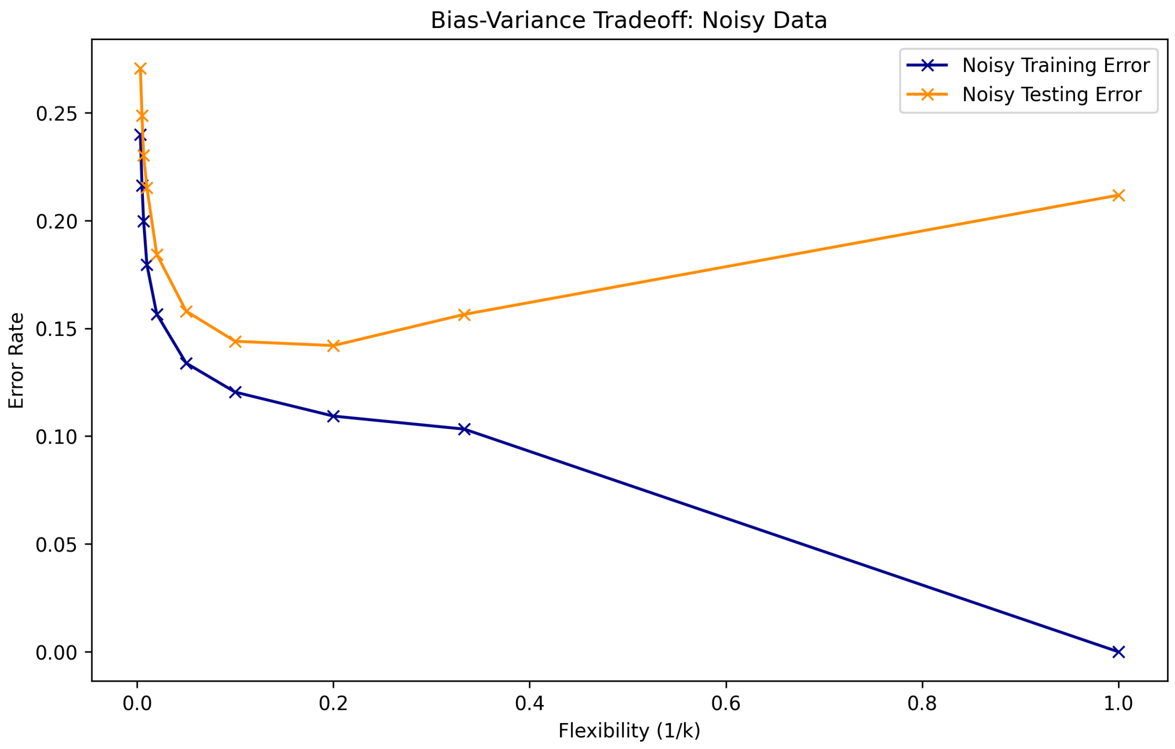
To investigate how the bias-variance tradeoff changes when the data is "messy," we will introduce random noise into the training and testing labels. We define a function `add_noise` that randomly selects 10% of the labels and replaces them with a random digit from 0 to 9.

```
Noise applied to 729 training samples.
Noise applied to 200 testing samples.
```

2.5 Re-evaluating kNN with Noisy Labels

Now, we will repeat the kNN classification process using the noisy labels and plot the training and testing error rates as a function of $1/k$.

3 Range of k: 1 to 300



Adding 10% label noise significantly altered the bias-variance tradeoff. While the original data allowed for high flexibility with minimal penalty, the noisy data shows a sharp increase in test error as $1/k$ increases. This demonstrates that in the presence of noise, simpler models (larger k) are preferred because they are more robust to outliers and incorrect labels.