Written Homework 1

Shubham Rastogi - sr7421
CS-GY 6923: Machine Learning

October 2, 2024

Problem 1: Minimizing a Weighted Loss Function.

Solution. (a) The weighted l_2 loss function is given by the equation:

$$L_{WSS}(\beta) = \sum_{i=1}^{n} \mathbf{w_i} \cdot (\langle \beta, \mathbf{x_i} \rangle - \mathbf{y_i})^2$$

We need to derive the expression for the gradient of $L_{WSS}(\beta)$.

First, lets express the loss function as a matrix:

$$L_{WSS}(\beta) = (X\beta - \mathbf{y})^T W (X\beta - \mathbf{y})$$

Where:

1. X is the input matrix of n samples
2. β is coefficient matrix for m+1 features
3. W is a diagonal matrix of weights for each sample
4. y is the output matrix

We can solve:

$$\nabla L_{WSS}(\beta) = \nabla (X\beta - \mathbf{y})^T W (X\beta - \mathbf{y})$$

by expanding the gradient,

$$\nabla L_{WSS}(\beta) = \begin{bmatrix} \frac{\partial L}{\partial \beta_1} \\ \vdots \\ \frac{\partial L}{\partial \beta_n} \end{bmatrix}$$

Solving the equation:

$$\frac{\partial L}{\partial \beta_i} = \lim_{\Delta \to 0} \frac{L(\beta + \Delta_{ei} - L(\beta)}{\Delta} \to (1)$$

Expanding,

$$L(\beta + \Delta_{ei}) - L(\beta)$$
$$= (X(\beta + \Delta_{ei}) - \mathbf{y})^T W (X(\beta + \Delta_{ei}) - \mathbf{y}) - (X\beta - \mathbf{y})^T W (X\beta - \mathbf{y})$$
$$= (X\beta + X\Delta_{ei} - \mathbf{y})^T W (X\beta + X\Delta_{ei} - \mathbf{y}) - (X\beta - \mathbf{y})^T W (X\beta - \mathbf{y})$$

Taking $X\Delta_{ei}$ as X_e^i and $X\beta - \mathbf{y}$ as C,

$$= (C + X_e^i)^T W (C + X_e^i) - C^T W C$$
$$= C^T W C + 2C^T W X_e^i + (X_e^i)^T W (X_e^i) - C^T W C$$
$$= 2C^T W X_e^i + (X_e^i)^T W (X_e^i) \rightarrow (2)$$

Applying (2) to (1) and expanding X_e^i,

$$\lim_{\Delta \to 0} \frac{2C^T W X \Delta_{ei} + (X\Delta_{ei})^T W (X\Delta_{ei})}{\Delta}$$

Removing the denominator Δ,

$$\lim_{\Delta \to 0} 2C^T W X + (X)^T W (X\Delta_{ei})$$

Solving for the limit we get,

$$2X^T W C = 2X^T W (X\beta - \mathbf{y})$$
$$\nabla L_{WSS}(\beta) = 2X^T W (X\beta - \mathbf{y}) \rightarrow (3)$$

□

Solution. (b) To find the closed form expression for $\beta^* = argmin_\beta L_{WSS}(\beta)$, we can equate the gradient of $L_{WSS}(\beta)$ to 0:

$$\nabla L_{WSS}(\beta) = 0$$
$$\Rightarrow 2X^T W (X\beta - \mathbf{y}) = 0$$
$$\Rightarrow X^T W X\beta - X^T W \mathbf{y} = 0$$
$$\Rightarrow X^T W X \beta = X^T W \mathbf{y}$$
$$\Rightarrow \beta = (X^T W X)^{-1} X^T W \mathbf{y} \rightarrow ClosedForm$$

Where W denotes the diagonal weight matrix:

$$W = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & w_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_n \end{bmatrix}$$

□

Problem 2: Machine Learning Does Averages.

Solution. (a) We are given the loss function:

$$L(m) = \sum_{i=1}^{n}(y_i - m)^2$$

We need to prove $L(m)_{min}$ exists when $m = \bar{y}$ where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ which is the **mean** of the data.

Expanding $L(m)$:

$$L(m) = \sum_{i=1}^{n}(y_i - m)^2$$

$$= \sum_{i=1}^{n}(y_i^2 - 2y_i m + m^2)$$

$$= \sum_{i=1}^{n} y_i^2 - 2m\sum_{i=1}^{n} y_i + nm^2)$$

Now $L(m)$ will be minimum when the differential is equal to 0. Applying the differential:

$$\frac{\partial L}{\partial m} = \frac{\partial}{\partial m}(\sum_{i=1}^{n} y_i^2 - 2m\sum_{i=1}^{n} y_i + nm^2))$$

$$= \frac{\partial}{\partial m}(\sum_{i=1}^{n} y_i^2) - \frac{\partial}{\partial m}(2m\sum_{i=1}^{n} y_i) + \frac{\partial}{\partial m}(nm^2)$$

$$= 0 - 2\sum_{i=1}^{n} y_i) + 2nm \rightarrow (1)$$

Equating (1) to 0,

$$\frac{\partial L_m in}{\partial m} = 0$$

$$2nm - 2\sum_{i=1}^{n} y_i) = 0$$

$$2nm = 2\sum_{i=1}^{n} y_i)$$

$$m = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Considering $\frac{1}{n}\sum_{i=1}^{n} y_i$ is equivalent to $\bar{y}$, L_{min} happens when $m = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ □

Solution. (b) We now study the value of m where $L(m)$ is minimum when $L(m) = max_i|y_i - m|$.

Consider y_i sorted:

$$y_1 < y_2 < y_3.... < y_n$$

We get y_n as y_{max} and y_1 as y_{min}.

Since $L(m)$ is most affected by the value with highest deviation from estimate, if we set m as either y_1 or y_n, we get the worst value for L, when $m = y_n$, $L(m) = max_i|y_i - y_n|$, which is maximum at $y_i = y_1$ (y_1 is furthest away from y_n).

Similarly, when $m = y_1$, $L(m) = max_i|y_i - y_1|$, which is maximum at $y_i = y_n$ (y_n is furthest away from y_1).

To minimise this error, we can set m as middle of y_1 and y_n, i.e., $L(m)$ is minimum when $m = (y_{min} + y_{max})/2$ □

Solution. (c) We now study that $L(m)$ is minimum for $L(m) = \sum_{i=1}^{n} |y_i - m|$ when m is median.

Consider y_i sorted- $y_1 < y_2 < y_3.... < y_n$.

Assuming n to be odd, and y_k is median, then the Loss,

$$L(y_k) = \sum_{i=1}^{n} |y_i - y_k| = \sum_{i=1}^{k-1} (y_k - y_i) + \sum_{i=k+1}^{n} (y_i - y_k)$$

Now, let's consider what happens if we choose a value slightly smaller or larger than the median:

If we choose $m = y_k - \epsilon$ (slightly smaller than median):

$$L(y_k - \epsilon) = \sum_{i=1}^{k-1} ((y_k - \epsilon) - y_i) + \sum_{i=k+1}^{n} (y_i - (y_k - \epsilon))$$

$$= \sum_{i=1}^{k-1} (y_k - y_i) - (k-1)\epsilon + \sum_{i=k+1}^{n} (y_i - y_k) + (n-k+1)\epsilon$$

$$= L(y_k) + (n - 2k + 2)\epsilon \text{ Larger than} L(y_k)$$

If we choose $m = y_k + \epsilon$ (slightly larger than median):

$$L(y_k + \epsilon) = \sum_{i=1}^{k-1} ((y_k + \epsilon) - y_i) + \sum_{i=k+1}^{n} (y_i - (y_k + \epsilon))$$

$$= \sum_{i=1}^{k-1} (y_k - y_i) + (k-1)\epsilon + \sum_{i=k+1}^{n} (y_i - y_k) - (n-k+1)\epsilon$$

$$= L(y_k) - (n - 2k + 2)\epsilon \text{ Larger than} L(y_k)$$

Since n is odd and y_k is the median, we know that $k = (n+1)/2$. Substituting this into our expressions:

$$L(y_k - \epsilon) = L(y_k) + (n - 2((n+1)/2) + 2)\epsilon = L(y_k) + \epsilon$$
$$L(y_k + \epsilon) = L(y_k) - (n - 2((n+1)/2) + 2)\epsilon = L(y_k) - \epsilon$$

This shows that moving slightly away from the median in either direction increases the loss.

For n is even with middle values y_k and y_{k+1} where $k = n/2$:

Loss function for $m \in [y_k, y_{k+1}]$:

$$L(m) = \sum_{i=1}^{n} |y_i - m| = \sum_{i=1}^{k} (m - y_i) + \sum_{i=k+1}^{n} (y_i - m)$$

Effect of small change ϵ within $[y_k, y_{k+1}]$:

$$L(m + \epsilon) = L(m) + k\epsilon - (n - k)\epsilon = L(m) + (2k - n)\epsilon$$

Since $k = n/2$, $2k - n = 0$, so:

$$L(m + \epsilon) = L(m)$$

This shows the loss function is constant within $[y_k, y_{k+1}]$.

Outside this range:

For $m < y_k$: Loss decreases as m increases

For $m > y_{k+1}$: Loss decreases as m decreases

Therefore, the loss function is minimized for any $m \in [y_k, y_{k+1}]$, this we can use $(y_k + y_{k+1})/2$ as the median for even-numbered datasets.

Therefore, the loss function $L(m)$ is minimized when m is the median. $\square$

Problem 3: Piecewise Linear Regression via Feature Transformations.

Solution. (a) We have the piecewise function,

$$f(x_i) = \begin{cases} a_1 + s_1 x_i & \text{for } x_i < \lambda \\ a_2 + s_2 x_i & \text{for } x_i \geq \lambda \rightarrow (1) \end{cases}$$

The constraint given here is that $a_1 + s_1\lambda = a_2 + s_2\lambda$. To convert this into an unconstrained model, we can obtain the following equation when $x_i = \lambda$,

$$a_1 + s_1\lambda = a_2 + s_2\lambda$$

$$a_2 = a_1 + s_1\lambda - s_2\lambda \rightarrow (2)$$

Substituting (2) in (1),

$$a_2 + s_2 x_i = a_1 + s_1\lambda - s_2\lambda + s_2 x_i, \text{ here } a_2 \text{ is unaffected by } x_i$$

Thus the unconstrained model becomes:

$$f(x_i) = \begin{cases} a_1 + s_1 x_i & \text{for } x_i < \lambda \\ a_1 + s_1\lambda - s_2\lambda + s_2 x_i & \text{for } x_i \geq \lambda \end{cases}$$

□

Solution. (b) We now need to find multiple linear regression fit for the function above. For this we first need to reduce the unconstrained piecewise function into a single function.

Take an Indicator matrix:

$$\mathbb{I} = \begin{cases} 0 & \text{for } x < \lambda \\ 1 & \text{for } x \geq \lambda \end{cases}$$

This gives us the equation:

$$f(x_i) = (a_1 + s_1 x_i)(1 - \mathbb{I}) + (a_1 + s_1\lambda - s_2\lambda + s_2 x_i)\mathbb{I}$$

$$= a_1 + s_1 x_i - a_1\mathbb{I} - s_1 x_i\mathbb{I} + a_1\mathbb{I} + s_1\lambda\mathbb{I} - s_2\lambda\mathbb{I} + s_2 x_i\mathbb{I}$$

$$= a_1 + s_1 x_i - s_1 x_i\mathbb{I} + s_1\lambda\mathbb{I} - s_2\lambda\mathbb{I} + s_2 x_i\mathbb{I}$$

$$= a_1 + s_1 x_i - \mathbb{I}(x_i - \lambda)(s_2 - s_1)$$

Taking $y = \beta_0 + \beta_1 x_1' + \beta_2 x_2'$,

$$\beta_0 = a_1, \beta_1 = s_1, \beta_2 = s_2 - s_1$$

or equivalently,

$$a_1 = \beta_0, s_1 = \beta_1, s_2 = \beta_2 + \beta_1$$

$$x'_0 = 1, x'_1 = x_i, x'_2 = \mathbb{I}(x_i - \lambda)$$

Data Matrix X will look like:

$$X = \begin{bmatrix} 1 & x_1 & \mathbb{I}(x_1 - \lambda) \\ 1 & x_2 & \mathbb{I}(x_2 - \lambda) \\ \vdots & \vdots & \vdots \\ 1 & x_n & \mathbb{I}(x_n - \lambda) \end{bmatrix}$$

The Optimal β can be found using the multiple linear regression equation:

$$\beta* = (X^T X)^-1 X^T \mathbf{y}$$

□

Solution. (c) The plots and code are discussed at the end of the file. □

Problem 4: Thinking About Data Transformations.

Solution. (a) Mean centering data columns in a multiple linear regression model does not improve the model's performance or loss:

Consider a multiple linear regression model with the following form,

$$\mathbf{y} = \beta_0 + \beta_1\mathbf{x_1} + \beta_2\mathbf{x_2} + \cdots + \beta_p\mathbf{x_p} + \varepsilon$$

Where,

- $\mathbf{y}$ is the dependent variable
- $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}$ are the independent variables
- $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients
- ε is the error

Applying mean centering to the independent variables (except for the first column of 1s),

$$\mathbf{x_i'} = \mathbf{x_i} - \mathbf{\bar{x}_i}$$

Where $\mathbf{\bar{x}_i}$ is the mean of the i-th column.

Substituting the mean-centered variables into our regression equation, we get,

$$\mathbf{y} = \beta_0' + \beta_1'(\mathbf{x_1} - \mathbf{\bar{x}_1}) + \beta_2'(\mathbf{x_2} - \mathbf{\bar{x}_2}) + \cdots + \beta_p'(\mathbf{x_p} - \mathbf{\bar{x}_p}) + \varepsilon$$

where β' are the new coefficients

Expanding this equation:

$$\mathbf{y} = \beta_0' + \beta_1'\mathbf{x_1} - \beta_1'\mathbf{\bar{x}_1} + \beta_2'\mathbf{x_2} - \beta_2'\mathbf{\bar{x}_2} + \cdots + \beta_p'\mathbf{x_p} - \beta_p'\mathbf{\bar{x}_p} + \varepsilon$$

Rearranging terms:

$$\mathbf{y} = (\beta_0' - \beta_1'\mathbf{\bar{x}_1} - \beta_2'\mathbf{\bar{x}_2} - \cdots - \beta_p'\mathbf{\bar{x}_p}) + \beta_1'\mathbf{x_1} + \beta_2'\mathbf{x_2} + \cdots + \beta_p'\mathbf{x_p} + \varepsilon$$

The coefficients for the x terms $(\beta_1', \beta_2', \ldots, \beta_p')$ remain the same as in the original model.

The $\bar{x}_i$ subtracted from the variables are constant.

The only change is in the intercept term, which becomes,

$$\beta_0 = \beta_0' - \beta_1'\mathbf{\bar{x}_1} - \beta_2'\mathbf{\bar{x}_2} - \cdots - \beta_p'\mathbf{\bar{x}_p}$$

equivalently, $\beta_1 = \beta_1', \beta_2 = \beta_2', ..., \beta_p = \beta_p'$

This change in the intercept does not affect the model's predictive power or its ability to fit the data.

Hence Mean centering the data columns in a multiple linear regression model only results in a shift of the intercept term and does not change the relationships between the independent variables and the dependent variable. Thereby it does not change the Loss □

Solution. (b) Consider the multiple linear regression model:

$$y = X\beta + \epsilon$$

Taking the normalizing matrix as a diagonal matrix:

$$S = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \frac{1}{\sigma_1^2} & 0 & \cdots & 0 \\ 1 & 0 & \frac{1}{\sigma_2^2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & \frac{1}{\sigma_n^2} \end{bmatrix}$$

Then the new input matrix X':

$$X' = XS^{-1}$$

New model:

$$\begin{aligned} y &= X'\beta' + \epsilon \\ &= XS^{-1}\beta' + \epsilon \end{aligned}$$

So β should be equal to $S^{-1}\beta$

Now consider the new β' equals to:

$$\beta' = (X'^T X')^{-1}(X'^T y)$$

Expanding X',

$$\begin{aligned} \beta' &= ((XS^{-1})^T(XS^{-1}))^{-1}((XS^{-1})^T y)) \\ &= S^T((X^TX)^{-1}SS^{-1}(X^Ty)) \\ &= S((X^TX)^{-1}(X^Ty)) \\ &= S\beta \end{aligned}$$

Which matches our initial finding. Thus normalizing the x values do not change the regression, they only change the scale of β and hence do not affect our loss □

Solution. (c) Effect of mean centering and normalisation on L_1 and L_∞ losses:

Consider $L_1 = \sum_{i=1}^n |y_i - x_i|$ loss, much like L_2 loss, the mean centering and normalization do not affect the performance of the loss function:

- Mean centering only shifts the data points by the means. It does not change the relationship between the points. The translation will not affect the loss.
- Normalization too will not affect the loss, since it only scales up/down the point values but does not change the relationship.

Now consider $L_\infty = max_i|y_i - x_i|$ loss, much like L_2 loss, the mean centering and normalization do not affect the performance of the loss function:

- Mean centering only shifts the data points by the means. Since the max difference will be the same, the loss remains the same.
- Normalization too will not affect the loss, since it only scales up/down the point values. The value of the max loss may scale up/down, but will still contribute to the error

In brief, mean centering and normalization fall under affine transformations, that may change the predictive line fit, but do not change the predictive power (since the points themselves are affected by the same transformations. $\square$

Problem 5: Student Question: Slightly More Efficient One-hot Encoding.

Solution. Consider a one-hot encoded matrix X with k categories and an intercept column (a column of ones). The intercept allows us to remove one of the one-hot columns without losing information.

The columns of X (including the intercept) are linearly dependent because the sum of all one-hot columns equals a vector of ones. Removing any one of these columns results in a set of linearly independent columns. This is shown by:

Let $X = [\mathbf{1}, X_1, X_2, \ldots, X_k]$ and assume we remove X_k, $\tilde{X} = [\mathbf{1}, X_1, X_2, \ldots, X_{k-1}]$.

The removed column can be expressed as a linear combination of the remaining columns and the intercept:

$$X_k = \mathbf{1} - (X_1 + X_2 + \ldots + X_{k-1})$$

This means that removing X_k does not change the span of the column space when an intercept is present.

Due to this property, removing a column still retains the total information provided by the one hot encoded columns, with the intercept + remaining columns giving information about the missing column. $\square$

Homework 1 Problem 3c

Loading the Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

The data for this demo comes from a survey of cars to determine the relation of mpg to engine characteristics. The data can be found in the UCI library: https://archive.ics.uci.edu/ml/datasets/auto+mpg. The specific files we need are in the "Data Folder" there: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg.

Import the data

```
names = ['mpg', 'cylinders','displacement', 'horsepower',
         'weight', 'acceleration', 'model year', 'origin', 'car name']

df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data',
                 header=None,delim_whitespace=True,names=names,na_values='?')

df.head(6)
```

```
/var/folders/7f/cmkbhd8j3zb25rcspw0bzl640000gn/T/ipykernel_4564/673634258.py:4: FutureWarning: The 'delim_whitespace' keywor
  df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data',
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car 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
5	15.0	8	429.0	198.0	4341.0	10.0	70	1	ford galaxie 500

```
df.dropna(inplace=True)
```

Set the target and features

Create the target y

```
y = df['mpg']
```

Create the features X

- X has 3 columns: column of 1's, horsepower, and $(horsepower - \lambda) * C$.
- C is 0 if horsepower is less than λ, otherwise C is 1.
- λ is 100

```
x = np.column_stack((np.ones(df.shape[0]), df['horsepower'], (df['horsepower'] - 100) * (df['horsepower'] > 100)))
```

Calculate the beta

$\beta = (X^T X)^{-1} X^T y$

```
beta = np.linalg.inv(np.dot(np.transpose(x), x)).dot(np.transpose(x)).dot(y)
print(beta)
```

```
[53.57724087 -0.32638817  0.234966  ]
```

Obtain the values of a_1, s_1 and s_2 and plot the data

$a_1 = \beta_0$ $s_1 = \beta_1$ $s_2 = \beta_2 + \beta_1$

and plot the function:

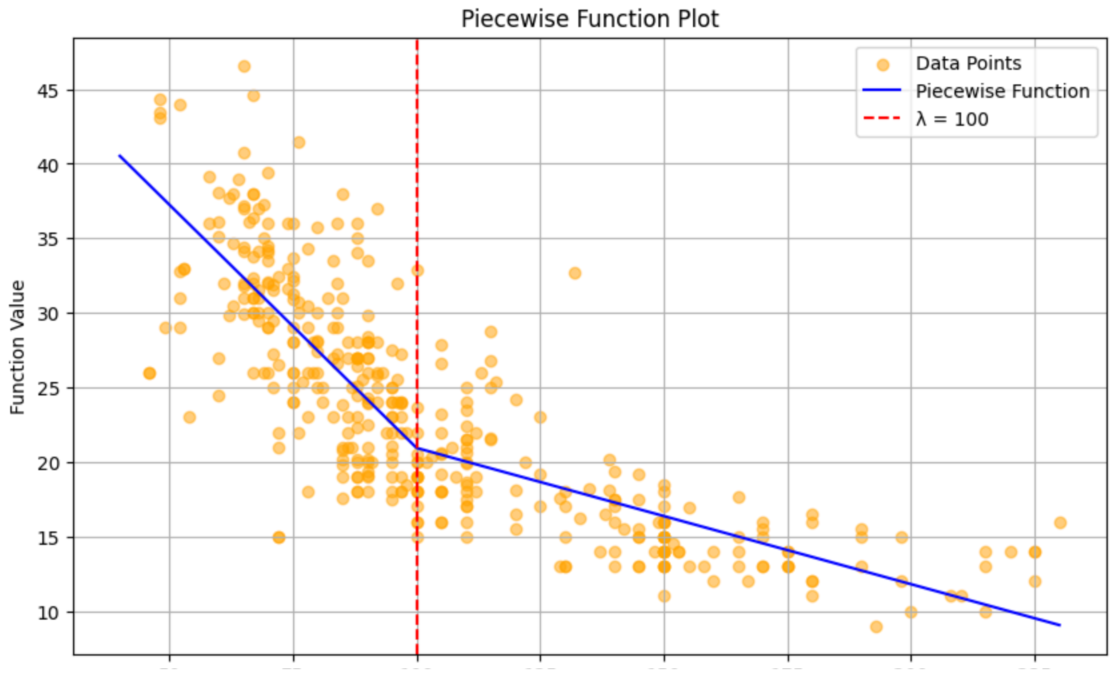
$$f(x_i) = \begin{cases} a_1 + s_1 x_i & \text{for } x_i < \lambda \\ a_1 + s_1\lambda + s_2(x_i - \lambda) & \text{for } x_i \geq \lambda \end{cases}$$

```
a1 = beta[0]
s1 = beta[1]
s2 = beta[2] + beta[1]


# Define the piecewise function
def piecewise_function(x):
    if x < 100:
        return a1 + s1 * x
    else:
        return a1 + s1 * 100 + s2 * (x - 100)

# Generate x values
x_values = np.linspace(40, 230, 400)
y_values = [piecewise_function(x) for x in x_values]

# Plot the piecewise function
plt.figure(figsize=(10, 6))
plt.scatter(df['horsepower'], df['mpg'], color='orange', label='Data Points', alpha=0.5)
plt.plot(x_values, y_values, label='Piecewise Function', color='blue')
plt.axvline(x=100, color='red', linestyle='--', label='λ = 100')
plt.title('Piecewise Function Plot')
plt.xlabel('Horsepower')
plt.ylabel('Function Value')
plt.legend()
plt.grid()
plt.show()
```


Piecewise Function Plot
Function Value
45
40
35
30
25
20
15
10
Data Points
Piecewise Function
λ = 100