Math 166: Statistics

Cooper Golemme

November 22, 2024

Question 1

(a) Fit a multiple linear regression model to predict MPG (miles per gallon) from the other variables. Summarize your analysis
Answer:

index	coef	std err	t	$P > \|t\|$
const	192.4377	23.5316	8.177	3.3510e-12
HP	0.3922	0.0814	4.8176	6.6715e-06
SP	-1.2948	0.2447	-5.2898	1.0191e-06
WT	-1.8598	0.2133	-8.7166	2.8888e-13
VOL	-0.0156	0.0228	-0.6854	0.4950

Table 1: Linear Regression to Predict MPG from the Other Variables

The table presents the results of a linear regression analysis to predict miles per gallon (MPG) based on various independent variables:

1. **VOL**: Cubic feet of cab space
2. **HP**: Engine horsepower
3. **SP**: Top speed (mph)
4. **WT**: Vehicle weight (100 lb)

Analysis of Results

- **Intercept (const):** The constant term in the model is 192.44, indicating the predicted MPG when all other variables are zero. This coefficient is highly statistically significant ($P < 0.001$).
- **Horsepower (HP):**
  - Coefficient: 0.3922
  - Interpretation: For every additional unit of horsepower, MPG is expected to increase by 0.3922, holding other variables constant.
  - Significance: Highly significant ($P = 6.67 \times 10^{-6}$).
- **Speed (SP):**
  - Coefficient: -1.2948
  - Interpretation: For every unit increase in speed, MPG decreases by 1.2948, holding other variables constant.
  - Significance: Highly significant ($P = 1.02 \times 10^{-6}$).
- **Weight (WT):**
  - Coefficient: -1.8598
  - Interpretation: For every unit increase in weight, MPG decreases by 1.8598, holding other variables constant.
  - Significance: Highly significant ($P = 2.89 \times 10^{-13}$).
- **Volume (VOL):**
  - Coefficient: -0.0156
  - Interpretation: For every unit increase in volume, MPG decreases by 0.0156, holding other variables constant.
  - Significance: Not statistically significant ($P = 0.4950$).

Conclusion

The most influential predictors of MPG are **weight (WT)** and **Top speed (SP)**, both showing large coefficients and very high statistical significance. **Horsepower (HP)** also significantly impacts MPG, though the effect is smaller in magnitude compared to SP and WT. **Volume (VOL)** has little to no impact on MPG, as its coefficient is small and not statistically significant.
This model suggests that reducing weight and speed might substantially improve fuel efficiency, while horsepower's positive impact is relatively minor. Volume seems unrelated to MPG based on this dataset.

Question 2

Part A:

We know that the kernal density estimator is:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-X_i}{h}\right)$$

To simplify notation, we can express

$$K_h(x, X_i) = \frac{1}{h}K\left(\frac{x-X_i}{h}\right)$$

So the $\hat{f}(x)$ can be written as

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}K_h(x, X_i)$$

We want to find $\mathbb{E}(\hat{f}(x))$ and we know that $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}K_h(x, X_i)$, so we first need to find, $\mathbb{E}(K_h(x, X_i))$

$$\mathbb{E}(K_h(x, X_i)) = \int \frac{1}{h}K\left(\frac{x-t}{h}\right)f(t)dt \tag{1}$$

$$= \frac{1}{h}\int \mathbb{1}(t-\frac{h}{2} < x < t+\frac{h}{2})f(t)dt \tag{2}$$

$$= \frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt \tag{3}$$

Thus,

$$\mathbb{E}(\hat{f}(x)) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}K_h(x, X_i)\right) = n^{-1}\sum_{i=1}^{n}\mathbb{E}(K_h(x, X_i))$$

Therefore,

$$\mathbb{E}(\hat{f}(x)) = \mathbb{E}(K_h(x, X_i)) = \frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt$$

Variance

In order to compute the variance of the kernel density estimator, we need to compute, $\mathbb{V}(K_h(x, X_i)$. It is important to note that the kernel only takes values 0 and 1, so $(K_h(x, X_i))^2 = \frac{1}{h}K_h(x, X_i)$.

$$\mathbb{V}(K_h(x, X_i)) = \mathbb{E}((K_h(x, X_i))^2) - \mathbb{E}(K_h(x, X_i))^2 \tag{4}$$

$$= \frac{1}{h}\mathbb{E}(K_h(x, X_i)) - \mathbb{E}(K_h(x, X_i))^2 \qquad \text{using the fact above} \tag{5}$$

$$= \frac{1}{h^2}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt - \left(\frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt\right)^2 \qquad \text{using the computed expectation} \tag{6}$$

We can now find the variance for the kernel density estimator:

$$\begin{aligned}
\mathbb{V}(\hat{f}(x)) &= \mathbb{V}(n^{-1}\sum_{i=1}^{n} K_h(x, X_i)) && (7)\\
&= \frac{1}{n^2}\sum_{i=1}^{n} \mathbb{V}(K_h(x, X_i)) && (8)\\
&= \frac{1}{n^2}\sum_{i=1}^{n} \frac{1}{h^2}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt - \left(\frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt\right)^2 && (9)\\
&= \frac{1}{n}\cdot\frac{1}{h^2}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt - \left(\frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt\right)^2 && (10)\\
&= \frac{1}{nh^2}\left[\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt - \left(\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt\right)^2\right] && (11)
\end{aligned}$$

Thus,

$$\mathbb{V}(\hat{f}(x)) = \frac{1}{nh^2}\left[\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt - \left(\int_{t-\frac{h}{2}}^{t+\frac{h}{2}} f(t)dt\right)^2\right]$$

Part B

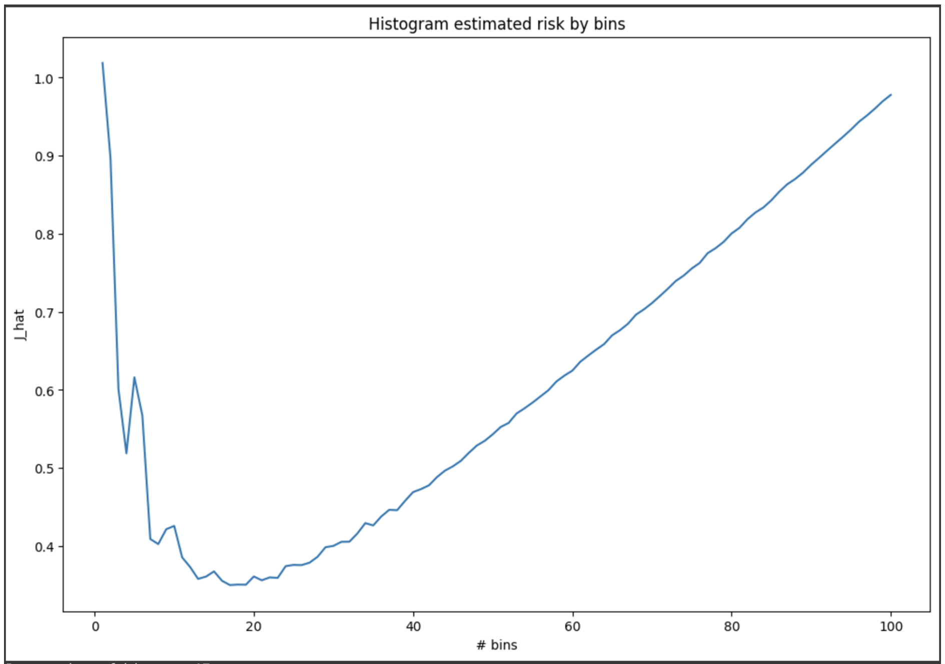
Claim: $h \to 0$ and $nh \to \infty$ as $n \to \infty$, then $\hat{f}(x) \xrightarrow{P} f(x)$

Proof. We know that the risk for this estimator is

$$R(f, \hat{f}_n) \approx \frac{1}{4}\sigma_K^4 h^4 \int (f''(x))^2 dx + \frac{\int K^2(x)dx}{nh}$$

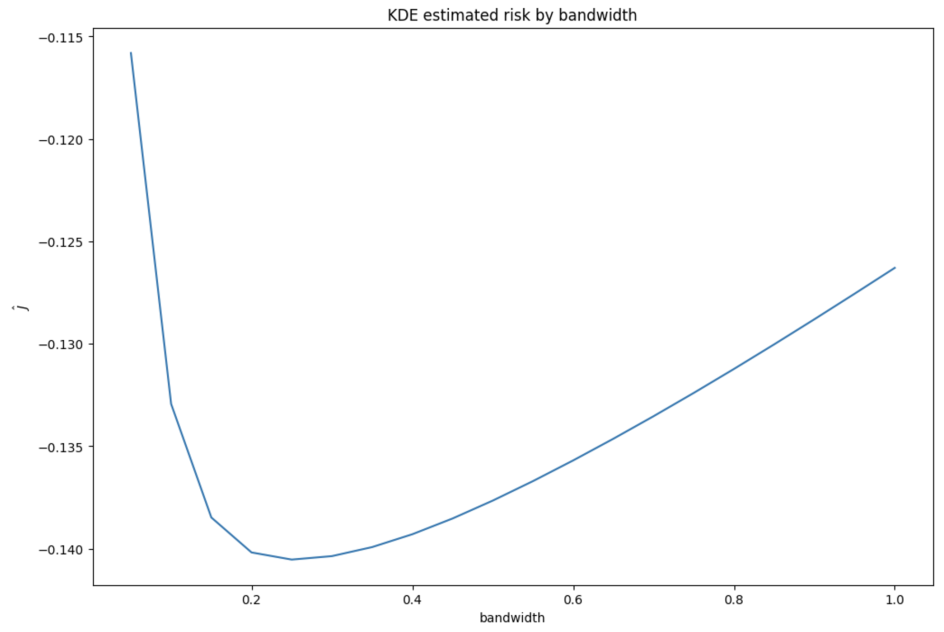
where $\sigma_K^2 = \int x^2 K(x)dx = \int_{-1/2}^{1/2} x^2 dx = 1/12$, and $\int K^2(x)dx = \int_{-1/2}^{1/2} dx = 1$. As $h \to 0$ and $nh \to \infty$, the first term goes to 0 and the second term also goes to 0, so the risk goes to 0.
The risk going to 0 implies that the sum of the bias squared and the variance goes to 0. Thus, a risk that converges to 0 means that f converges in probability to $\hat{f}$ as the bias for the estimator converges to 0. □

Question 3



```
Best number of bins:    17
Risk with 17 bins:      0.350
```

Figure 1: Risk by Number of Bins



```
Best bandwidth:                0.252
Risk with selected bandwidth:  −0.141
```

Figure 2: Comparing Bandwidth

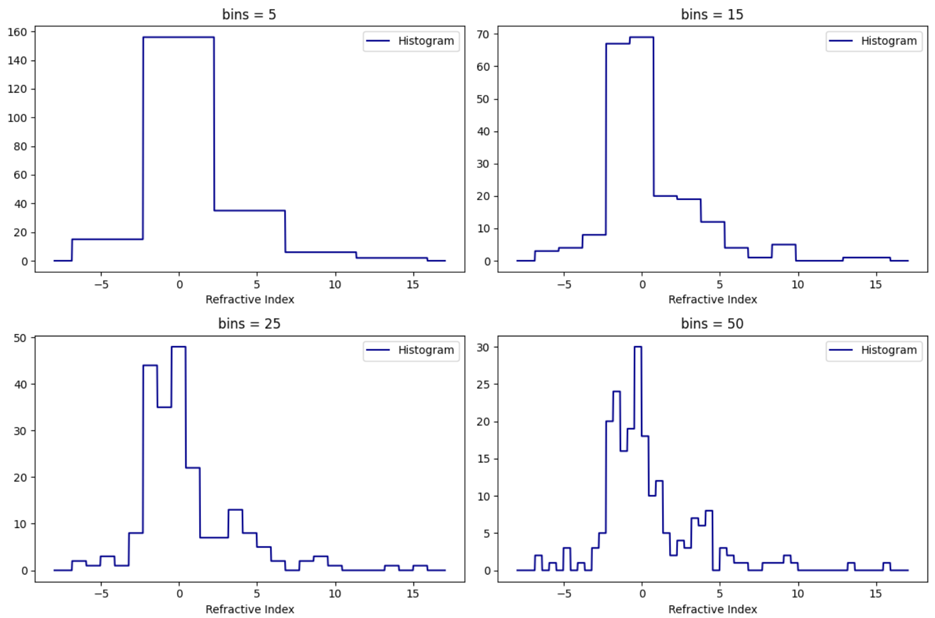


Figure 3: Comparing Bin Numbers

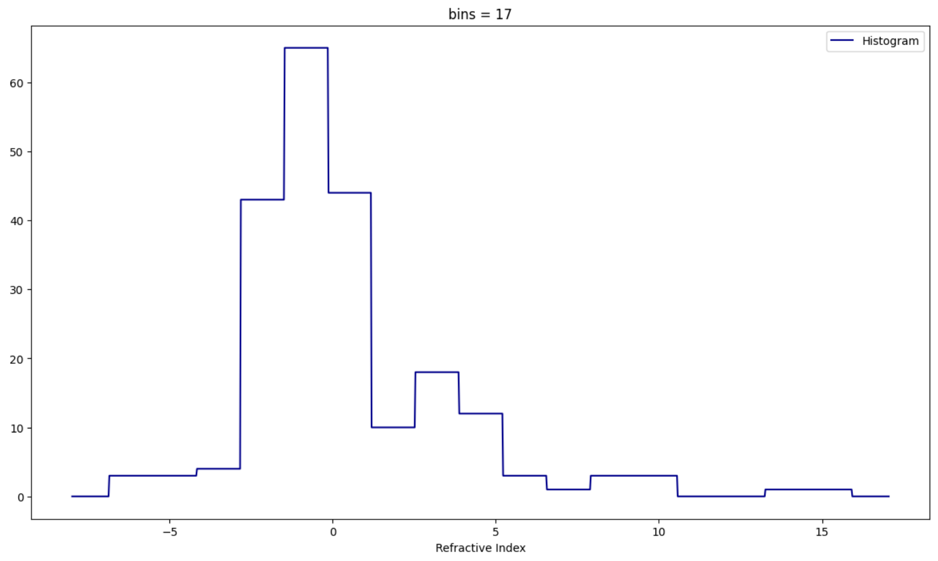


Figure 4: Best 17 Bins

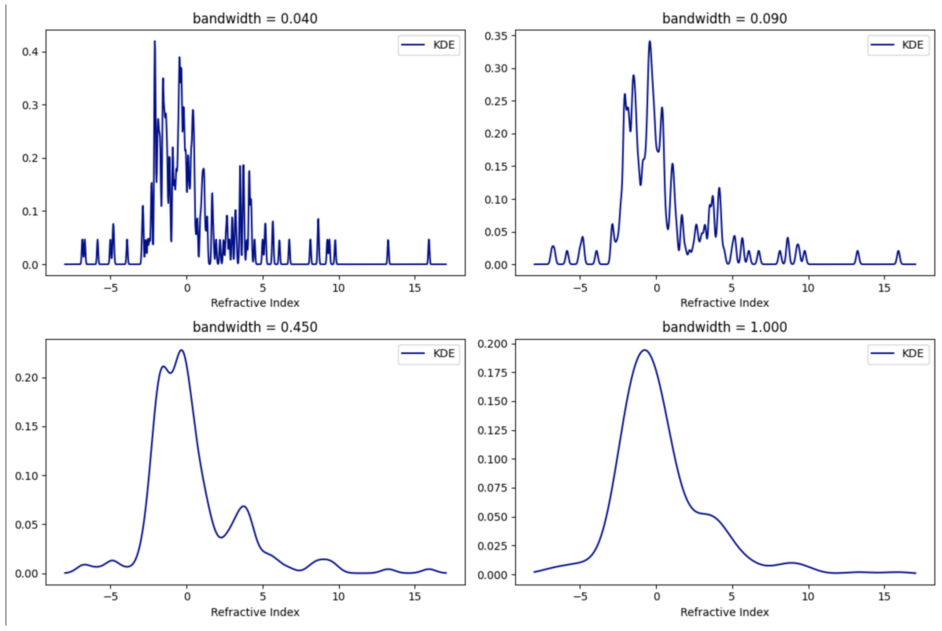


Figure 5: Comparing using different bandwidths

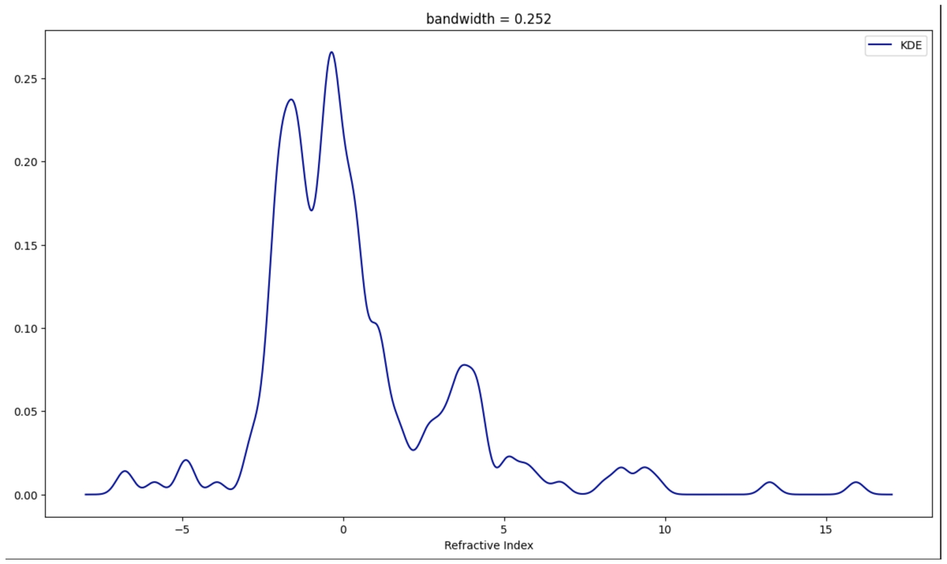


Figure 6: Using the optimal bandwidth

Similarities: For each different bin number and bandwidth value, the shape of the resulting graph is roughly the same. There is clustering around 0 and some peaks round 3, 10. and smaller ones around 15.
Differences: The bandwidth and the number of bins have inverse effects on the resulting graphs. As we increase the number of bins, the histogram becomes less smooth and more jagged. The smaller the widths of the bins the greater variation is possible between the number of elements in each bin. As we decrease the bandwidth, we notice a similar result of

increased jaggedness. Smaller bandwidth values result in greater changes in the graph value from point to point.
Comments on Best Fit: Greater number of bins and lower bandwidth values fits the data better than the alternatives, but we run the risk of over-fitting, to the point where we are just connecting the points on the scatter plot and not generating meaningful predictive functions. We can see this by calculating the risk for each bin number and bandwidth values as seen in Figure 1 and Figure 2. Both exhibit a trend were too small values have high risk and too large values have high risk. Somewhere in the middle, the risk is minimized. This aligns with the observations we find when generating the plots for various bandwidths and bin numbers. Too smooth or too jagged functions smooth out variations in the data or give too much weight to small variations respectively. This is emblematic of a general trend we can observe with regression.

Supplemental Question

(a) When $d > n$

$d > n$ means that the number of parameters exceeds the number of observations. It makes sense therefore, that we can not run regression to get a singular function because we have some "free" variables that we have not observed.
More mathematically it means that the matrix X has linearly dependent columns and thus $X^T X$ is not invertible.
Thus, $X^T X\beta = X^T y$ has infinitely many solutions because the null space of X, $null(X) \neq \{0\}$. The problem with having finitely many solutions to this equation is that there is not one singular curve of best fit for linear regression; there are infinitely many to choose from, and without a good way to choose, we run the risk of over fitting the training data, which causes generalization problems

(b)

$$\hat{\beta} = \underset{\beta\in\mathbb{R}^{d\times 1}}{\arg\min} ||y - X\beta||_2^2 + \lambda||\beta||_2^2 \tag{12}$$

$$= \underset{\beta\in\mathbb{R}^{d\times 1}}{\arg\min}(y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta \tag{13}$$

$$\tag{14}$$

To find the arg min, we can differentiate with respect to β

$$\frac{\partial}{\partial\beta}(y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta = 0 \tag{15}$$

$$= -2X^T y + 2X^T X\beta + 2\lambda\beta = 0 \tag{16}$$

$$= -X^T y + X^T X\beta + \lambda\beta \tag{17}$$

$$X^T X\beta + \lambda\beta = X^T y \tag{18}$$

$$(X^T X + I\lambda)\beta = X^T y \tag{19}$$

$$\beta = X^T y(X^T X + I\lambda)^{-1} \tag{20}$$

Thus, the optimal value for β is $X^T y(X^T X + I\lambda)^{-1}$

(c) Discussion

The problem with linear regression in the case where $d > n$ is that $X^T X\beta = X^T y$ has infinitely many solutions. We discussed this observation in part (a). This stems from the observation that if $d > n$, then $X^T X$ has linearly independent rows and is not invertible. We can get a better linear regression in the case where $d > n$ by adding this tunable parameter λ to the optimization problem.
As we showed in part (b), the optimal value for $\beta = X^T y(X^T X + I\lambda)^{-1}$. When we add $I\lambda$ to $X^T X$ in the part inside the parenthesis, $X^T X + I\lambda$, we can ensure that the resulting

$(X^TX + I\lambda)$ is invertible. This has to do with the properties of symmetric matrices and positive definite.
X^TX is not invertible so it can have some zero Eigenvalues. The effect of adding the $d \times d$ identity matrix times a $\lambda > 0$ is that the resulting matrix's Eigenvalues are all positive and thus invertible.
The property of $(X^TX + I\lambda)$ being invertible, ensures that there is a unique solution to the equation $\beta = X^Ty(X^TX + I\lambda)^{-1}$, which makes regression possible.

Code for Question 1 and Question 3

All of the code is available in a Google Colab project here.

Question 1:

Getting and Cleaning Data

```
import pandas as pd
import requests
import re

def float_or_int(x):
    try:
        return int(x)
    except:
      return float(x)

# Download the data from the URL
url = "http://www.stat.cmu.edu/~larry/all-of-statistics/=data/carmileage.dat"
response = requests.get(url)
response.raise_for_status()  # Raise an exception for bad status codes

# Read the data into a Pandas DataFrame
data = response.text
data = data[data.find("MAKE"):]
data = re.sub('\t+', ' ', data)
data = re.sub(' +', ' ', data)
data = data.replace("\t", ",")
data = data.replace(" ", ",")

# replace multiple spaces with 1
lines = data.split('\n')

lines = [line for line in lines if line != ""]
# print(lines)

header = lines[0].split(',')
header[0] = header[0] + "/" + header[1]
header.pop(1)
data_rows = []
for i in range(1, len(lines)):
    row = lines[i].split(',')
    if (len(row) > 6):
      row[0] = row[0] + "/" + row[1]
```

```
          row.pop(1)
        row0 = row[0]
        row = [float_or_int(x) for x in row[1:]]
        row.insert(0, row0)
        # print(row)
        data_rows.append(row)

df = pd.DataFrame(data_rows, columns=header)

df.head(100)
```

Regression

```
import numpy as np
import pandas as pd
from scipy.stats import t

def regression(X, Y):
    X = X.copy()
    X = X.apply(pd.to_numeric, errors='coerce')
    X = X.dropna()

    # Create new column with all 1s for intercept at start
    X.insert(0, 'const', 1)

    beta_hat = (np.linalg.inv(X.T @ X) @ X.T @ Y).to_numpy()

    Y_pred = X @ beta_hat

    epsilon_hat = Y_pred - Y

    training_error = epsilon_hat.T @ epsilon_hat

    sigma2_hat = (training_error / (Y.shape[0] - X.shape[1]))

    se_beta_hat = np.sqrt(sigma2_hat * np.diag(np.linalg.inv(X.T @ X))).T

    t_values = beta_hat.reshape(-1) / se_beta_hat

    p_values = 2 * (1 - t.cdf(np.abs(t_values), X.shape[0] - 1))

    return pd.DataFrame({
        'coef': beta_hat.reshape(-1),
        'std err': se_beta_hat.reshape(-1),
        't': t_values.reshape(-1),
        'P > |t|': p_values.reshape(-1)
        }, index=X.columns)
```

Running Model

```
features = ['HP', 'SP', 'WT', 'VOL']

Y = df['MPG']
X = df[features]

regression(X, Y)
```

Question 3:

Get and Clean Data

```
import requests
import pandas as pd
import re

urlglass = "http://www.stat.cmu.edu/~larry/all-of-statistics/=data/glass.dat"

response = requests.get(urlglass)

d = response.text

rows = d.split('\n')

parsed_rows = [re.sub(r'\s+', ' ', row) for row in rows]
parsed_rows = [row.split(" ") for row in parsed_rows]
parsed_rows[0][0] = "to_remove"

df = pd.DataFrame(parsed_rows[1:], columns=parsed_rows[0])

df = df.drop(columns=["to_remove"])

df.head(100)

df.dropna(inplace=True)
X = df['RI']
X = X.astype(float)
```

Functions to Generate and Plot Data

```
import numpy as np
import pandas as pd
from itertools import product
from scipy.stats import norm


def rescale(X):
    X_min, X_max = X.min(), X.max()
    if X_max == X_min:
        return X - X_min
    return (X - X_min) / (X_max - X_min)

def j_hat_histogram(X, m):
    n = len(X)
    h = 1 / m
    xx = rescale(X)
    phat = np.array([np.sum(np.where((j / m <= xx) & (xx < (j + 1) / m), 1, 0)) for j in
    phat[-1] += np.sum(np.where(xx == 1, 1, 0))
    phat = phat / n
    return 2 / ((n - 1) * h) + (n + 1) / (n - 1) * np.sum(phat **2)

def j_hat_kde(X, h):
    n = len(X)
    # print(X)
    Kstar_args = np.array([X.iloc[i] - X.iloc[j] for i, j in product(range(n), range(n))
    sum_value = np.sum(norm.pdf(Kstar_args) - 2 * norm.pdf(Kstar_args))
    return sum_value / (h * n * n) + 2 * norm.pdf(0) / (n * h)

def create_kde(X, bandwidth, alpha):
    n = len(X)

    def get_x(t):
        XX = np.repeat(X.to_numpy().reshape(-1, 1), len(t), axis=1)
        tt = np.repeat(t.reshape(1, -1), n, axis=0)
        return np.sum(norm.pdf((tt - XX) / bandwidth), axis=0) / (n * bandwidth)

    def get_values(t):
        return get_x(t)

    return get_values

def create_histogram(X, m, alpha):
    n = len(X)
    X_min, X_max = X.min(), X.max()
    xx = rescale(X)
    phat = np.array([np.sum(np.where((j / m <= xx) & (xx < (j + 1) / m), 1, 0)) for j in
    phat[-1] += np.sum(np.where(xx == 1, 1, 0))
    z = norm.ppf(1 - alpha / (2 * m))
    c = z * np.sqrt(m / n) / 2

    def get_bin(t):
        return np.maximum(np.minimum(np.floor(m * (t - X_min) / (X_max - X_min)).astype(

    def get_values(t):
        x = np.where((t >= X_min) & (t <= X_max), phat[get_bin(t)], 0)
        # lower = (np.maximum(np.sqrt(x) - c, 0))**2
        # upper = (np.sqrt(x) + c)**2
        return x

    return get_values

def plot_histogram(X, bins, ax):
    X_min, X_max = X.min(), X.max()
    x_plot_vals = np.arange(X_min - 0.05 * (X_max - X_min), X_max + 0.05 * (X_max - X_mi
    vals = create_histogram(X, m=bins, alpha=0.05)(x_plot_vals)
    ax.plot(x_plot_vals, vals, color='darkblue', label='Histogram')
    ax.legend()

    # Title and labels
    ax.set_title('bins = %d' % bins)
    ax.set_xlabel('Refractive Index')
```

Bin Plotting

```
import matplotlib.pyplot as plt
%matplotlib inline

# Calculate and plot estimated risk for various numbers of bins
m_values = [m for m in range(1, 101)]
j_hat = [j_hat_histogram(X, m) for m in m_values]

best_m = m_values[np.argmin(j_hat)]
best_risk = min(j_hat)

plt.figure(figsize=(12, 8))
plt.plot(m_values, j_hat)
plt.xlabel('# bins')
plt.ylabel('J_hat')
plt.title('Histogram estimated risk by bins')
plt.show()

print('Best number of bins:\t%i' % best_m)
print('Risk with %i bins: \t%.3f' % (best_m, best_risk))
```

Bandwidth Plotting Code

```
import matplotlib.pyplot as plt
%matplotlib inline

# Calculate and plot estimated risk for various bandwidths
```

```
h_values = [m / 20 for m in range(1, 21)]
j_hat = [j_hat_kde(X, h) for h in h_values]

from scipy.optimize import minimize

res = minimize(fun = lambda h: j_hat_kde(X, h), x0 = 0.2, options={'maxiter': 10})

best_h = res.x[0]
best_risk = res.fun

plt.figure(figsize=(12, 8))
plt.plot(h_values, j_hat)
plt.xlabel('bandwidth')
plt.ylabel(r'$\hat{J}$')
plt.title('KDE estimated risk by bandwidth')
plt.show()

print('Best bandwidth:\t\t\t%.3f' % best_h)
print('Risk with selected bandwidth: \t%.3f' % best_risk)
```

Bins Histogram Code

```
# Show 4 different bin counts
plt.figure(figsize=(12, 8))
for i, bins in enumerate([5, 15, 25, 50]):

    # Set up the plot
    ax = plt.subplot(2, 2, i + 1)
    plot_histogram(X, bins, ax)

plt.tight_layout()
plt.show()

# Best risk
plt.figure(figsize=(14.5, 8))
plot_histogram(X, bins=17, ax=plt.gca())
plt.show()
```

Bandwidth Plotting Code

```
plt.figure(figsize=(12, 8))
for i, bandwidth in enumerate([0.04, 0.09, 0.45, 1.0]):
    # Set up the plot
    ax = plt.subplot(2, 2, i + 1)
    plot_kde(X, bandwidth, ax)
```

```
plt.tight_layout()
plt.show()

# Best risk
plt.figure(figsize=(14.5, 8))
plot_kde(X, bandwidth=best_h, ax=plt.gca())
plt.show()
```