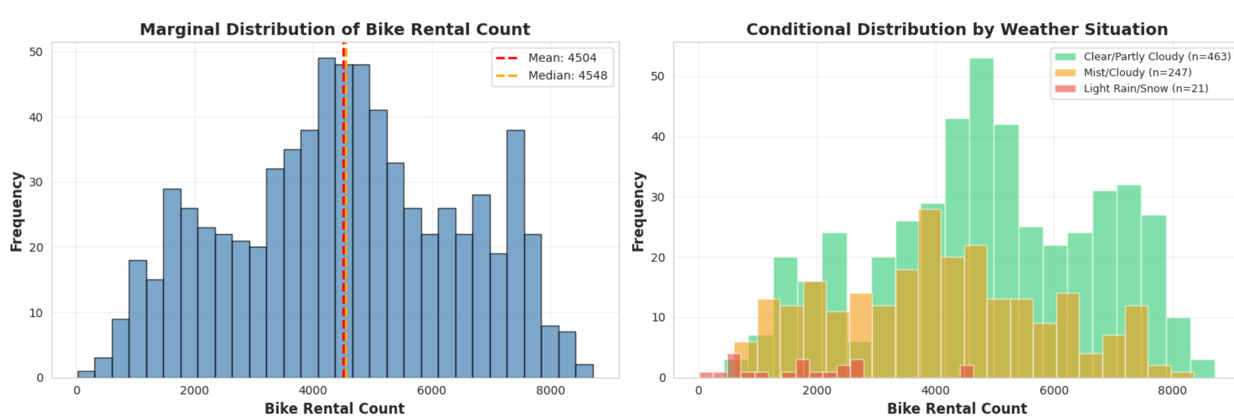
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import f1_score

# Set plotting style
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (15, 5)

# ===== Question 1(a): Plot Distributions =====
print("="*60)
print("Question 1(a): Marginal and Conditional Distributions")
print("="*60)

# Load data
df =
pd.read_csv(r'/content/drive/MyDrive/Bike-Sharing-Dataset/day.csv')

# Create figure with two subplots
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

# Left plot: Marginal distribution of cnt
axes[0].hist(df['cnt'], bins=30, edgecolor='black', alpha=0.7,
color='steelblue')
axes[0].set_xlabel('Bike Rental Count', fontsize=12,
fontweight='bold')
axes[0].set_ylabel('Frequency', fontsize=12, fontweight='bold')
axes[0].set_title('Marginal Distribution of Bike Rental Count',
fontsize=14, fontweight='bold')
axes[0].axvline(df['cnt'].mean(), color='red', linestyle='--',
linewidth=2,
                label=f'Mean: {df["cnt"].mean():.0f}')
axes[0].axvline(df['cnt'].median(), color='orange', linestyle='--',
linewidth=2,
                label=f'Median: {df["cnt"].median():.0f}')
axes[0].legend(fontsize=10)
axes[0].grid(True, alpha=0.3)

# Right plot: Conditional distribution by weather situation
weather_labels = {
```

```
    1: 'Clear/Partly Cloudy',
    2: 'Mist/Cloudy',
    3: 'Light Rain/Snow',
    4: 'Heavy Rain/Snow'
}

colors = ['#2ecc71', '#f39c12', '#e74c3c', '#8e44ad']

for idx, weather in enumerate(sorted(df['weathersit'].unique())):
    subset = df[df['weathersit'] == weather]['cnt']
    label = weather_labels.get(weather, f'Weather {weather}')
    axes[1].hist(subset, bins=20, alpha=0.6,
                 label=f'{label} (n={len(subset)})',
                 color=colors[idx] if idx < len(colors) else None)

axes[1].set_xlabel('Bike Rental Count', fontsize=12,
fontweight='bold')
axes[1].set_ylabel('Frequency', fontsize=12, fontweight='bold')
axes[1].set_title('Conditional Distribution by Weather Situation',
fontsize=14, fontweight='bold')
axes[1].legend(fontsize=9)
axes[1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('q1a_distributions.png', dpi=300, bbox_inches='tight')
plt.show()
```

```
============================================================
Question 1(a): Marginal and Conditional Distributions
============================================================
```




```
# ===== Question 1(b): Linear Regression =====
print("="*60)
print("Question 1(b): Linear Regression")
print("="*60)
```

```python
# Modeling approach: Treat weathersit as categorical variable
# Use dummy variables with weathersit=1 as reference category
X_weather = pd.get_dummies(df['weathersit'], prefix='weather',
drop_first=True)
y = df['cnt']

print("\nModeling Approach:")
print("- weathersit is CATEGORICAL (not numeric)")
print("- Use DUMMY VARIABLES (one-hot encoding)")
print("- Reference category: weathersit=1 (Clear weather)")
print(f"- Dummy columns created: {X_weather.columns.tolist()}")

# Fit model
model = LinearRegression()
model.fit(X_weather, y)

# Report coefficients
print("\n" + "-"*60)
print("Model Coefficients:")
print("-"*60)
print(f"Intercept: {model.intercept_:.2f}")
print(f"  → Expected count for weathersit=1 (Clear)")

for i, col in enumerate(X_weather.columns):
    weather_num = col.split('_')[1]
    print(f"\n{col}: {model.coef_[i]:.2f}")
    print(f"  → Difference from Clear to weathersit={weather_num}")

# Expected counts by weather
print("\n" + "-"*60)
print("Expected Rental Counts:")
print("-"*60)
print(f"weathersit=1 (Clear): {model.intercept_:.2f}")
for i, col in enumerate(X_weather.columns):
    weather_num = int(col.split('_')[1])
    expected = model.intercept_ + model.coef_[i]
    print(f"weathersit={weather_num}: {expected:.2f}")

print("\n" + "="*60)
```

```
============================================================
Question 1(b): Linear Regression
============================================================

Modeling Approach:
- weathersit is CATEGORICAL (not numeric)
- Use DUMMY VARIABLES (one-hot encoding)
- Reference category: weathersit=1 (Clear weather)
- Dummy columns created: ['weather_2', 'weather_3']
```

```
------------------------------------------------------------
Model Coefficients:
------------------------------------------------------------
Intercept: 4876.79
  → Expected count for weathersit=1 (Clear)

weather_2: -840.92
  → Difference from Clear to weathersit=2

weather_3: -3073.50
  → Difference from Clear to weathersit=3

------------------------------------------------------------
Expected Rental Counts:
------------------------------------------------------------
weathersit=1 (Clear): 4876.79
weathersit=2: 4035.86
weathersit=3: 1803.29

============================================================
```

```
# ===== Question 1(c): Difference between Clear (1) and Wet (3) =====
print("="*60)
print("Question 1(c): Expected Ride Count Difference")
print("="*60)

# Create dummy variables (same as part b)
X_weather = pd.get_dummies(df['weathersit'], prefix='weather',
drop_first=True)
y = df['cnt']

# Fit model
model = LinearRegression()
model.fit(X_weather, y)

# Calculate expected counts
expected_clear = model.intercept_  # weathersit=1 (reference)

# Find coefficient for weathersit=3
if 'weather_3' in X_weather.columns:
    coef_3_index = X_weather.columns.tolist().index('weather_3')
    expected_wet = model.intercept_ + model.coef_[coef_3_index]

    difference = expected_clear - expected_wet

    print("\nExpected rental counts:")
    print(f"  Clear weather (weathersit=1): {expected_clear:.2f}
bikes")
    print(f"  Wet weather (weathersit=3):   {expected_wet:.2f} bikes")
    print(f"\nDifference: {difference:.2f} bikes")
```

```
    print(f"\nInterpretation:")
    print(f"Clear weather is expected to have {difference:.2f} more 
rentals")
    print(f"than wet weather (light rain/snow).")
else:
    print("\nNote: weathersit=3 not found in data or has no 
observations.")

print("\n" + "="*60)
```

```
============================================================
Question 1(c): Expected Ride Count Difference
============================================================

Expected rental counts:
  Clear weather (weathersit=1): 4876.79 bikes
  Wet weather (weathersit=3):   1803.29 bikes

Difference: 3073.50 bikes

Interpretation:
Clear weather is expected to have 3073.50 more rentals
than wet weather (light rain/snow).

============================================================
```

```
# ===== Question 1(d): Model Evaluation Metrics =====
print("="*60)
print("Question 1(d): RSS, R², and Residual Standard Error")
print("="*60)

# Use the model from part (b)
y_pred = model.predict(X_weather)
residuals = y - y_pred

# 1. Residual Sum of Squares (RSS)
RSS = np.sum(residuals**2)

# 2. Total Sum of Squares (TSS)
TSS = np.sum((y - y.mean())**2)

# 3. R² (Coefficient of Determination)
R2 = 1 - (RSS / TSS)

# 4. Residual Standard Error
n = len(y)
p = X_weather.shape[1]  # number of predictors
residual_std_error = np.sqrt(RSS / (n - p - 1))

print("\n" + "-"*60)
print("Model Evaluation Metrics:")
```

```
print("-"*60)
print(f"Residual Sum of Squares (RSS): {RSS:.2f}")
print(f"Total Sum of Squares (TSS):    {TSS:.2f}")
print(f"R² (R-squared):                {R2:.4f}")
print(f"Residual Standard Error:       {residual_std_error:.2f}")
```

```
============================================================
Question 1(d): RSS, R², and Residual Standard Error
============================================================

------------------------------------------------------------
Model Evaluation Metrics:
------------------------------------------------------------
Residual Sum of Squares (RSS): 2467890819.44
Total Sum of Squares (TSS):    2739535392.05
R² (R-squared):                0.0992
Residual Standard Error:       1841.18
```

```
# ===== Question 1(e): Multiple Linear Regression =====
print("="*60)
print("Question 1(e): Multiple Linear Regression (All Weather
Variables)")
print("="*60)

# Prepare feature matrix: weathersit as dummy + continuous variables
X_weather_dummy = pd.get_dummies(df['weathersit'], prefix='weather',
drop_first=True)
X_all = pd.concat([X_weather_dummy, df[['temp', 'hum', 'windspeed']]],
axis=1)

# Fit multiple regression model
model_all = LinearRegression()
model_all.fit(X_all, y)

# Temperature impact analysis
print("\n" + "-"*60)
print("Temperature Impact Analysis:")
print("-"*60)

temp_coef = model_all.coef_[X_all.columns.tolist().index('temp')]
print(f"\nTemperature coefficient: {temp_coef:.2f}")
print(f"\nFor 10°C increase in actual temperature:")
print(f"  - Normalized temp increase: 10/41 = {10/41:.4f}")
print(f"  - Expected count increase: {temp_coef:.2f} × {10/41:.4f} =
{temp_coef * (10/41):.2f} bikes")

print(f"\nInterpretation:")
print(f"A 10-degree Celsius increase in temperature is associated
with")
print(f"an expected increase of {temp_coef * (10/41):.2f} bike
```

```
rentals,")
print(f"holding other variables constant.")

print("\n" + "="*60)
```

```
============================================================
Question 1(e): Multiple Linear Regression (All Weather Variables)
============================================================

------------------------------------------------------------
Temperature Impact Analysis:
------------------------------------------------------------

Temperature coefficient: 6395.16

For 10°C increase in actual temperature:
  - Normalized temp increase: 10/41 = 0.2439
  - Expected count increase: 6395.16 × 0.2439 = 1559.79 bikes

Interpretation:
A 10-degree Celsius increase in temperature is associated with
an expected increase of 1559.79 bike rentals,
holding other variables constant.

============================================================
```

```
# ===== Question 1(f): Logistic Regression (threshold=4000) =====
print("="*60)
print("Question 1(f): Logistic Regression with threshold=4000")
print("="*60)



# Create binary labels: Low Demand (≤4000) vs High Demand (>4000)
threshold = 4000
df['demand'] = (df['cnt'] > threshold).astype(int)

print(f"\nThreshold: {threshold}")
print(f"Low Demand (cnt ≤ {threshold}):  {(df['demand']==0).sum()}
samples ({(df['demand']==0).sum()/len(df)*100:.1f}%)")
print(f"High Demand (cnt > {threshold}): {(df['demand']==1).sum()}
samples ({(df['demand']==1).sum()/len(df)*100:.1f}%)")

# Prepare features: weathersit as dummy + continuous variables
X_weather_dummy = pd.get_dummies(df['weathersit'], prefix='weather',
drop_first=True)
X = pd.concat([X_weather_dummy, df[['temp', 'hum', 'windspeed']]],
axis=1)
y_binary = df['demand']

# Train-test split
```

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y_binary, test_size=0.3, random_state=42
)

print(f"\nTrain set size: {len(X_train)}")
print(f"Test set size:  {len(X_test)}")



log_model = LogisticRegression(max_iter=1000, random_state=42)
log_model.fit(X_train, y_train)

# Predictions
y_train_pred = log_model.predict(X_train)
y_test_pred = log_model.predict(X_test)

# Evaluate
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print("\n" + "-"*60)
print("Model Performance:")
print("-"*60)
print(f"Training accuracy: {train_accuracy:.4f}")
print(f"Test accuracy:     {test_accuracy:.4f}")
```

```
============================================================
Question 1(f): Logistic Regression with threshold=4000
============================================================

Threshold: 4000
Low Demand (cnt ≤ 4000):  279 samples (38.2%)
High Demand (cnt > 4000): 452 samples (61.8%)

Train set size: 511
Test set size:  220

------------------------------------------------------------
Model Performance:
------------------------------------------------------------
Training accuracy: 0.8395
Test accuracy:     0.8045
```

```python
# ===== Question 1(g): Logistic Regression (custom threshold) =====
print("="*60)
print("Question 1(g): Logistic Regression with Custom Threshold")
print("="*60)

# Choose criterion: Use MEDIAN for balanced classes
new_threshold = df['cnt'].median()
```

```
print("\nThreshold Selection Criterion:")
print("Using MEDIAN to create balanced classes")
print("- Median naturally splits data into two equal-sized groups")
print("- Balanced classes help avoid model bias toward majority
class")
print("- Improves model's ability to learn both classes equally")

print(f"\nNew threshold (median): {new_threshold:.0f}")

# Create new binary labels
df['demand_new'] = (df['cnt'] > new_threshold).astype(int)

print(f"\nClass distribution with new threshold:")
print(f"Low Demand (cnt ≤ {new_threshold:.0f}):
{(df['demand_new']==0).sum()} samples
({(df['demand_new']==0).sum()/len(df)*100:.1f}%)")
print(f"High Demand (cnt > {new_threshold:.0f}):
{(df['demand_new']==1).sum()} samples
({(df['demand_new']==1).sum()/len(df)*100:.1f}%)")

# Compare with previous threshold
print(f"\nComparison with previous threshold ({threshold}):")
print(f"Previous - Low: {(df['demand']==0).sum()}
({(df['demand']==0).sum()/len(df)*100:.1f}%), High:
{(df['demand']==1).sum()} ({(df['demand']==1).sum()/len(df)*100:.1f}
%)")
print(f"New      - Low: {(df['demand_new']==0).sum()}
({(df['demand_new']==0).sum()/len(df)*100:.1f}%), High:
{(df['demand_new']==1).sum()}
({(df['demand_new']==1).sum()/len(df)*100:.1f}%)")
print("→ New threshold creates more balanced classes")

# Train-test split with new labels
y_binary_new = df['demand_new']
X_train_g, X_test_g, y_train_g, y_test_g = train_test_split(
    X, y_binary_new, test_size=0.3, random_state=42
)

# Fit logistic regression
log_model_g = LogisticRegression(max_iter=1000, random_state=42)
log_model_g.fit(X_train_g, y_train_g)

# Predictions
y_train_pred_g = log_model_g.predict(X_train_g)
y_test_pred_g = log_model_g.predict(X_test_g)

# Evaluate
train_accuracy_g = accuracy_score(y_train_g, y_train_pred_g)
test_accuracy_g = accuracy_score(y_test_g, y_test_pred_g)
train_f1_g = f1_score(y_train_g, y_train_pred_g)
```

```python
test_f1_g = f1_score(y_test_g, y_test_pred_g)

print("\n" + "-"*60)
print("Model Performance (New Threshold):")
print("-"*60)
print(f"Training accuracy: {train_accuracy_g:.4f}")
print(f"Test accuracy:     {test_accuracy_g:.4f}")
print(f"Training F1-score: {train_f1_g:.4f}")
print(f"Test F1-score:     {test_f1_g:.4f}")


# Also calculate metrics for previous model (f)
train_f1_f = f1_score(y_train, y_train_pred)
test_f1_f = f1_score(y_test, y_test_pred)

# Comparison table
print("\n" + "="*60)
print("COMPARISON: Model (f) vs Model (g)")
print("="*60)
print(f"\n{'Metric':<20} {'Model (f)':<15} {'Model (g)':<15} {'Difference':<15}")
print("-"*65)
print(f"{'Threshold':<20} {threshold:<15} {new_threshold:<15.0f} {'-':<15}")
print(f"{'Test Accuracy':<20} {test_accuracy:<15.4f} {test_accuracy_g:<15.4f} {test_accuracy_g-test_accuracy:+.4f}")
print(f"{'Test F1-score':<20} {test_f1_f:<15.4f} {test_f1_g:<15.4f} {test_f1_g-test_f1_f:+.4f}")

print("\n" + "-"*60)
print("Performance Analysis:")
print("-"*60)


print(f"✗ Accuracy decreased by {(test_accuracy-test_accuracy_g)*100:.2f}%")
print(f"✗ F1-score decreased by {(test_f1_f-test_f1_g)*100:.2f}%")

print("\n" + "-"*60)
print("Rationale for Performance Change:")
print("-"*60)

print("The threshold=4000 may better capture true 'high demand':")
print("- Higher threshold creates more meaningful separation")
print("- May align better with business definition of 'high demand'")
print("- Trade-off between statistical balance and practical relevance")
```

```
============================================================
Question 1(g): Logistic Regression with Custom Threshold
```

```
============================================================

Threshold Selection Criterion:
Using MEDIAN to create balanced classes
- Median naturally splits data into two equal-sized groups
- Balanced classes help avoid model bias toward majority class
- Improves model's ability to learn both classes equally

New threshold (median): 4548

Class distribution with new threshold:
Low Demand (cnt ≤ 4548):  366 samples (50.1%)
High Demand (cnt > 4548): 365 samples (49.9%)

Comparison with previous threshold (4000):
Previous - Low: 279 (38.2%), High: 452 (61.8%)
New      - Low: 366 (50.1%), High: 365 (49.9%)
→ New threshold creates more balanced classes

------------------------------------------------------------
Model Performance (New Threshold):
------------------------------------------------------------
Training accuracy: 0.7750
Test accuracy:     0.8000
Training F1-score: 0.7826
Test F1-score:     0.7800

============================================================
COMPARISON: Model (f) vs Model (g)
============================================================

Metric               Model (f)       Model (g)       Difference
-----------------------------------------------------------------
Threshold            4000            4548            -
Test Accuracy        0.8045          0.8000          -0.0045
Test F1-score        0.8352          0.7800          -0.0552

------------------------------------------------------------
Performance Analysis:
------------------------------------------------------------
✗ Accuracy decreased by 0.45%
✗ F1-score decreased by 5.52%

------------------------------------------------------------
Rationale for Performance Change:
------------------------------------------------------------
The threshold=4000 may better capture true 'high demand':
- Higher threshold creates more meaningful separation
- May align better with business definition of 'high demand'
- Trade-off between statistical balance and practical relevance
```

2025Fall_CS526_HW2

xc166

October 2025

1 Question 2

(a): Mean of Residuals is Zero

To Prove:

In simple linear regression, the mean of the residuals e_i is always zero:

$$\bar{e} = \frac{1}{n}\sum_{i=1}^{n} e_i = 0$$

Proof:

Step 1: Define the residual

The residual for observation i is:

$$e_i = Y_i - \hat{Y}_i$$

where $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$ is the predicted value.

Step 2: Sum the residuals

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

Substitute $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$:

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n}[Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i)]$$

$$= \sum_{i=1}^{n} Y_i - \sum_{i=1}^{n} \hat{\beta}_0 - \sum_{i=1}^{n} \hat{\beta}_1 X_i$$

$$= \sum_{i=1}^{n} Y_i - n\hat{\beta}_0 - \hat{\beta}_1 \sum_{i=1}^{n} X_i$$

Step 3: Use the normal equation

In simple linear regression, the least squares estimates satisfy the **first normal equation**:

$$\sum_{i=1}^{n}(Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i) = 0$$

This is equivalent to:

$$\sum_{i=1}^{n} Y_i = n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} X_i$$

Step 4: Substitute back

From Step 2, we have:

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n} Y_i - n\hat{\beta}_0 - \hat{\beta}_1 \sum_{i=1}^{n} X_i$$

Using the normal equation from Step 3:

$$\sum_{i=1}^{n} e_i = 0$$

Step 5: Calculate the mean

$$\bar{e} = \frac{1}{n}\sum_{i=1}^{n} e_i = \frac{1}{n} \cdot 0 = 0$$

(b): Residuals are Orthogonal to Predictor

To Prove:

In simple linear regression, the residuals e_i are orthogonal to the predictor variable X_i:

$$\sum_{i=1}^{n} X_i e_i = 0$$

Proof:

Step 1: Define the residual

The residual for observation i is:

$$e_i = Y_i - \hat{Y}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i)$$

Step 2: Compute the dot product

$$\sum_{i=1}^{n} X_i e_i = \sum_{i=1}^{n} X_i(Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)$$

Expand:

$$= \sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \hat{\beta}_0 - \sum_{i=1}^{n} X_i \hat{\beta}_1 X_i$$

$$= \sum_{i=1}^{n} X_i Y_i - \hat{\beta}_0 \sum_{i=1}^{n} X_i - \hat{\beta}_1 \sum_{i=1}^{n} X_i^2$$

Step 3: Use the second normal equation

In simple linear regression, the least squares estimates satisfy the **second normal equation**:

$$\sum_{i=1}^{n} X_i (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i) = 0$$

This is obtained by taking the partial derivative of $\sum e_i^2$ with respect to $\hat{\beta}_1$ and setting it to zero:

$$\frac{\partial}{\partial \hat{\beta}_1} \sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2 = -2 \sum_{i=1}^{n} X_i (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i) = 0$$

This gives us:

$$\sum_{i=1}^{n} X_i Y_i = \hat{\beta}_0 \sum_{i=1}^{n} X_i + \hat{\beta}_1 \sum_{i=1}^{n} X_i^2$$

Step 4: Substitute back

From Step 2:

$$\sum_{i=1}^{n} X_i e_i = \sum_{i=1}^{n} X_i Y_i - \hat{\beta}_0 \sum_{i=1}^{n} X_i - \hat{\beta}_1 \sum_{i=1}^{n} X_i^2$$

Using the second normal equation from Step 3:

$$\sum_{i=1}^{n} X_i e_i = 0$$

(c): Residuals Uncorrelated with Predicted Response

To Prove:

In simple linear regression, the residuals are uncorrelated with the predicted responses:

$$\text{Cov}(e, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} (e_i - \bar{e})(\hat{Y}_i - \bar{\hat{Y}}) = 0$$

Proof:

Step 1: Simplify using result from part (a)

From part (a), we know that $\bar{e} = 0$.
Therefore:

$$\text{Cov}(e, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(e_i - 0)(\hat{Y}_i - \bar{\hat{Y}})$$

$$= \frac{1}{n}\sum_{i=1}^{n} e_i(\hat{Y}_i - \bar{\hat{Y}})$$

Step 2: Expand the predicted value

Recall that $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$.
Also, $\bar{\hat{Y}} = \frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i = \frac{1}{n}\sum_{i=1}^{n}(\hat{\beta}_0 + \hat{\beta}_1 X_i) = \hat{\beta}_0 + \hat{\beta}_1 \bar{X}$
Therefore:

$$\hat{Y}_i - \bar{\hat{Y}} = (\hat{\beta}_0 + \hat{\beta}_1 X_i) - (\hat{\beta}_0 + \hat{\beta}_1 \bar{X})$$

$$= \hat{\beta}_1(X_i - \bar{X})$$

Step 3: Substitute into covariance

$$\text{Cov}(e, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n} e_i \cdot \hat{\beta}_1(X_i - \bar{X})$$

$$= \frac{\hat{\beta}_1}{n}\sum_{i=1}^{n} e_i(X_i - \bar{X})$$

$$= \frac{\hat{\beta}_1}{n}\left[\sum_{i=1}^{n} e_i X_i - \sum_{i=1}^{n} e_i \bar{X}\right]$$

$$= \frac{\hat{\beta}_1}{n}\left[\sum_{i=1}^{n} e_i X_i - \bar{X}\sum_{i=1}^{n} e_i\right]$$

Step 4: Apply results from parts (a) and (b)

From part (a): $\sum_{i=1}^{n} e_i = 0$
From part (b): $\sum_{i=1}^{n} X_i e_i = 0$
Therefore:

$$\text{Cov}(e, \hat{Y}) = \frac{\hat{\beta}_1}{n}[0 - \bar{X} \cdot 0] = 0$$

(d): Mean of Predicted Equals Mean of Observed

To Prove:

In simple linear regression, the mean of the predicted responses equals the mean of the observed responses:

$$\bar{\hat{Y}} = \bar{Y}$$

Proof:

Step 1: Express the relationship between Y, $\hat{Y}$, and e

For each observation:

$$Y_i = \hat{Y}_i + e_i$$

Step 2: Sum both sides

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \hat{Y}_i + \sum_{i=1}^{n} e_i$$

Step 3: Apply result from part (a)

From part (a), we know that $\sum_{i=1}^{n} e_i = 0$.
Therefore:

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \hat{Y}_i + 0$$

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \hat{Y}_i$$

Step 4: Divide by n to get means

$$\frac{1}{n}\sum_{i=1}^{n} Y_i = \frac{1}{n}\sum_{i=1}^{n} \hat{Y}_i$$

$$\bar{Y} = \bar{\hat{Y}}$$

(e): Proof that $R^2 = \frac{ESS}{TSS}$

To Prove:

Starting from the definition $R^2 = 1 - \frac{RSS}{TSS}$, show that:

$$R^2 = \frac{ESS}{TSS}$$

where:

- $RSS = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ (Residual Sum of Squares)
- $TSS = \sum_{i=1}^{n}(Y_i - \bar{Y})^2$ (Total Sum of Squares)
- $ESS = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2$ (Explained Sum of Squares)

Proof:

Step 1: Decompose the total deviation

For each observation, we can write:

$$Y_i - \bar{Y} = (\hat{Y}_i - \bar{Y}) + (Y_i - \hat{Y}_i)$$

$$Y_i - \bar{Y} = (\hat{Y}_i - \bar{Y}) + e_i$$

Step 2: Square both sides and sum

$$\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \sum_{i=1}^{n}[(\hat{Y}_i - \bar{Y}) + e_i]^2$$

Expand the right side:

$$= \sum_{i=1}^{n}[(\hat{Y}_i - \bar{Y})^2 + 2(\hat{Y}_i - \bar{Y})e_i + e_i^2]$$

$$= \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 + 2\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})e_i + \sum_{i=1}^{n}e_i^2$$

Step 3: Show the cross-product term equals zero

We need to show that $\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})e_i = 0$.

From part (d), we know $\bar{\hat{Y}} = \bar{Y}$.

Therefore:

$$\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})e_i = \sum_{i=1}^{n}(\hat{Y}_i - \bar{\hat{Y}})e_i$$

From part (c), we proved that:

$$\sum_{i=1}^{n}(\hat{Y}_i - \bar{\hat{Y}})e_i = n \cdot \mathrm{Cov}(e, \hat{Y}) = n \cdot 0 = 0$$

Step 4: Establish TSS = ESS + RSS

From Step 2 and Step 3:

$$\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 + \sum_{i=1}^{n}e_i^2$$

$$TSS = ESS + RSS$$

Step 5: Derive $R^2 = \frac{ESS}{TSS}$

From the definition:

$$R^2 = 1 - \frac{RSS}{TSS}$$

From Step 4, we have $RSS = TSS - ESS$, so:

$$R^2 = 1 - \frac{TSS - ESS}{TSS}$$

$$= 1 - \frac{TSS}{TSS} + \frac{ESS}{TSS}$$

$$= 1 - 1 + \frac{ESS}{TSS}$$

$$= \frac{ESS}{TSS}$$

(f): Proof that $R^2 = r_{XY}^2$

To Prove:

In simple linear regression with Y as the response and X as the predictor, the R^2 statistic equals the square of the correlation coefficient:

$$R^2 = r_{XY}^2$$

where:

$$r_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2 \sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

Proof:

Step 1: Recall the formula for $\hat{\beta}_1$
In simple linear regression, the slope is:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

Step 2: Express R^2 using part (e)
From part (e), we know:

$$R^2 = \frac{ESS}{TSS} = \frac{\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$

Step 3: Simplify the numerator (ESS)
From part (c), we showed that:

$$\hat{Y}_i - \bar{Y} = \hat{\beta}_1 (X_i - \bar{X})$$

Therefore:

$$ESS = \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2 = \sum_{i=1}^n [\hat{\beta}_1 (X_i - \bar{X})]^2$$

$$= \hat{\beta}_1^2 \sum_{i=1}^{n} (X_i - \bar{X})^2$$

Step 4: Substitute into R^2

$$R^2 = \frac{\hat{\beta}_1^2 \sum_{i=1}^{n} (X_i - \bar{X})^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

Step 5: Substitute the formula for $\hat{\beta}_1$

From Step 1:

$$\hat{\beta}_1^2 = \left[\frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (X_i - \bar{X})^2} \right]^2$$

$$= \frac{[\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})]^2}{[\sum_{i=1}^{n} (X_i - \bar{X})^2]^2}$$

Substitute into Step 4:

$$R^2 = \frac{[\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})]^2}{[\sum_{i=1}^{n} (X_i - \bar{X})^2]^2} \cdot \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

$$= \frac{[\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})]^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2 \cdot \sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

Step 6: Recognize the correlation coefficient

The expression above is exactly:

$$R^2 = \left[\frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2 \sum_{i=1}^{n} (Y_i - \bar{Y})^2}} \right]^2 = r_{XY}^2$$

2 Question 3: Logistic Regression Manual Calculations

Given Data:

Sample	Sugar Intake (x)	Condition (y)
1	30	0
2	50	0
3	70	1
4	90	1

Initial parameters: $\theta_0 = 0$, $\theta_1 = 0$

(a) Calculate $h_\theta(x)$ for each training example

The sigmoid function is:

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 \cdot x)}}$$

With $\theta_0 = 0$ and $\theta_1 = 0$:

$$h_\theta(x) = \frac{1}{1 + e^{-(0+0 \cdot x)}} = \frac{1}{1 + e^0} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

Calculations:
Sample 1: $x^{(1)} = 30$

$$h_\theta(30) = 0.5$$

Sample 2: $x^{(2)} = 50$

$$h_\theta(50) = 0.5$$

Sample 3: $x^{(3)} = 70$

$$h_\theta(70) = 0.5$$

Sample 4: $x^{(4)} = 90$

$$h_\theta(90) = 0.5$$

Result: All samples have $h_\theta(x^{(i)}) = 0.5$ because the initial parameters are both zero.

(b) Calculate the log-likelihood $\ell(\theta)$

The log-likelihood function is:

$$\ell(\theta) = \sum_{i=1}^{4} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Since $h_\theta(x^{(i)}) = 0.5$ for all samples:

- $\log(0.5) = \log(1/2) = -\log(2) \approx -0.6931$
- $\log(1 - 0.5) = \log(0.5) = -\log(2) \approx -0.6931$

Calculations by sample:
Sample 1: $x^{(1)} = 30, y^{(1)} = 0$

$$\ell_1 = 0 \cdot \log(0.5) + (1 - 0) \cdot \log(0.5) = \log(0.5) = -\log(2)$$

Sample 2: $x^{(2)} = 50, y^{(2)} = 0$

$$\ell_2 = 0 \cdot \log(0.5) + (1 - 0) \cdot \log(0.5) = \log(0.5) = -\log(2)$$

Sample 3: $x^{(3)} = 70, y^{(3)} = 1$

$$\ell_3 = 1 \cdot \log(0.5) + (1 - 1) \cdot \log(0.5) = \log(0.5) = -\log(2)$$

Sample 4: $x^{(4)} = 90, y^{(4)} = 1$

$$\ell_4 = 1 \cdot \log(0.5) + (1-1) \cdot \log(0.5) = \log(0.5) = -\log(2)$$

Total log-likelihood:

$$\ell(\theta) = \ell_1 + \ell_2 + \ell_3 + \ell_4 = 4 \cdot (-\log(2)) = -4\log(2)$$

$$\ell(\theta) = -4 \times 0.6931 \approx -2.7726$$

(c) First iteration of gradient ascent

Gradient Formulas:

$$\frac{\partial \ell(\theta)}{\partial \theta_0} = \sum_{i=1}^{4} (y^{(i)} - h_\theta(x^{(i)}))$$

$$\frac{\partial \ell(\theta)}{\partial \theta_1} = \sum_{i=1}^{4} (y^{(i)} - h_\theta(x^{(i)})) \cdot x^{(i)}$$

Calculate gradients:
For θ_0:

$$\frac{\partial \ell}{\partial \theta_0} = (0-0.5) + (0-0.5) + (1-0.5) + (1-0.5)$$

$$= -0.5 - 0.5 + 0.5 + 0.5 = 0$$

For θ_1:

$$\frac{\partial \ell}{\partial \theta_1} = (0-0.5) \cdot 30 + (0-0.5) \cdot 50 + (1-0.5) \cdot 70 + (1-0.5) \cdot 90$$

$$= (-0.5)(30) + (-0.5)(50) + (0.5)(70) + (0.5)(90)$$

$$= -15 - 25 + 35 + 45 = 40$$

Update parameters with $\alpha = 0.01$:
Update rule:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \alpha \frac{\partial \ell}{\partial \theta_j}$$

For θ_0:

$$\theta_0^{\text{new}} = 0 + 0.01 \times 0 = 0$$

For θ_1:

$$\theta_1^{\text{new}} = 0 + 0.01 \times 40 = 0.4$$

Updated parameters after first iteration:

$$\theta_0 = 0, \quad \theta_1 = 0.4$$

Explanation of the process:
Gradient ascent is used to maximize the log-likelihood:

1. Start with initial parameters
2. Calculate the gradient (direction of steepest increase)
3. Update parameters in the direction of the gradient: $\theta_j := \theta_j + \alpha \frac{\partial \ell}{\partial \theta_j}$
4. Repeat until convergence

The gradient tells us that increasing θ_1 will increase the log-likelihood, which makes sense because higher sugar intake is associated with higher probability of developing the condition.

3 Question 4: Multiclass Perceptron Manual Calculations

Given Data:

Training samples:

Sample	x_1 (Excitement)	x_2 (Budget)	True Label (y)
1	3	100	0
2	8	300	1
3	5	150	2

Initial weight matrix:

$$W = \begin{bmatrix} 0.4 & 0.1 & -0.3 \\ 0.3 & -0.2 & 0.5 \\ -0.1 & 0.3 & 0.2 \end{bmatrix}$$

Row 1 $\rightarrow$ Category 0, Row 2 $\rightarrow$ Category 1, Row 3 $\rightarrow$ Category 2
Column 1 $\rightarrow$ bias, Column 2 $\rightarrow$ x_1 coefficient, Column 3 $\rightarrow$ x_2 coefficient

(a) Update weights for each training sample

Sample 1: $x_1 = 3, x_2 = 100, y = 0$

Step 1: Form input vector (with bias)

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ 3 \\ 100 \end{bmatrix}$$

Step 2: Calculate scores for each category
Category 0:

$$\text{score}_0 = 0.4 \cdot 1 + 0.1 \cdot 3 + (-0.3) \cdot 100 = 0.4 + 0.3 - 30 = -29.3$$

Category 1:

$$\text{score}_1 = 0.3 \cdot 1 + (-0.2) \cdot 3 + 0.5 \cdot 100 = 0.3 - 0.6 + 50 = 49.7$$

Category 2:

$$\text{score}_2 = (-0.1) \cdot 1 + 0.3 \cdot 3 + 0.2 \cdot 100 = -0.1 + 0.9 + 20 = 20.8$$

Step 3: Predict category

$$\hat{y}^{(1)} = \arg\max(\text{score}_0, \text{score}_1, \text{score}_2) = \arg\max(-29.3, 49.7, 20.8) = 1$$

Step 4: Check if correct
True label: $y^{(1)} = 0$, Predicted: $\hat{y}^{(1)} = 1$
Prediction is WRONG! $\rightarrow$ Update weights
Step 5: Update weights
Update rule:

- $\mathbf{w}_{\text{true}} := \mathbf{w}_{\text{true}} + \mathbf{x}$ (increase true category)
- $\mathbf{w}_{\text{predicted}} := \mathbf{w}_{\text{predicted}} - \mathbf{x}$ (decrease predicted category)

Increase row 0 (true category):

$$\mathbf{w}_0^{\text{new}} = [0.4, 0.1, -0.3] + [1, 3, 100] = [1.4, 3.1, 99.7]$$

Decrease row 1 (predicted category):

$$\mathbf{w}_1^{\text{new}} = [0.3, -0.2, 0.5] - [1, 3, 100] = [-0.7, -3.2, -99.5]$$

Row 2 unchanged:

$$\mathbf{w}_2^{\text{new}} = [-0.1, 0.3, 0.2]$$

Updated weight matrix after Sample 1:

$$W = \begin{bmatrix} 1.4 & 3.1 & 99.7 \\ -0.7 & -3.2 & -99.5 \\ -0.1 & 0.3 & 0.2 \end{bmatrix}$$

Sample 2: $x_1 = 8, x_2 = 300, y = 1$

Step 1: Form input vector

$$\mathbf{x}^{(2)} = \begin{bmatrix} 1 \\ 8 \\ 300 \end{bmatrix}$$

Step 2: Calculate scores (using updated weights)
Category 0:

$$\text{score}_0 = 1.4 \cdot 1 + 3.1 \cdot 8 + 99.7 \cdot 300 = 1.4 + 24.8 + 29910 = 29936.2$$

Category 1:

$$\text{score}_1 = (-0.7) \cdot 1 + (-3.2) \cdot 8 + (-99.5) \cdot 300 = -0.7 - 25.6 - 29850 = -29876.3$$

Category 2:

$$\text{score}_2 = (-0.1) \cdot 1 + 0.3 \cdot 8 + 0.2 \cdot 300 = -0.1 + 2.4 + 60 = 62.3$$

Step 3: Predict category

$$\hat{y}^{(2)} = \arg\max(29936.2, -29876.3, 62.3) = 0$$

Step 4: Check if correct
True label: $y^{(2)} = 1$, Predicted: $\hat{y}^{(2)} = 0$
Prediction is WRONG! → Update weights
Step 5: Update weights
Increase row 1 (true category):

$$\mathbf{w}_1^{\text{new}} = [-0.7, -3.2, -99.5] + [1, 8, 300] = [0.3, 4.8, 200.5]$$

Decrease row 0 (predicted category):

$$\mathbf{w}_0^{\text{new}} = [1.4, 3.1, 99.7] - [1, 8, 300] = [0.4, -4.9, -200.3]$$

Row 2 unchanged:

$$\mathbf{w}_2^{\text{new}} = [-0.1, 0.3, 0.2]$$

Updated weight matrix after Sample 2:

$$W = \begin{bmatrix} 0.4 & -4.9 & -200.3 \\ 0.3 & 4.8 & 200.5 \\ -0.1 & 0.3 & 0.2 \end{bmatrix}$$

Sample 3: $x_1 = 5, x_2 = 150, y = 2$

Step 1: Form input vector

$$\mathbf{x}^{(3)} = \begin{bmatrix} 1 \\ 5 \\ 150 \end{bmatrix}$$

Step 2: Calculate scores (using updated weights)
Category 0:

$$\text{score}_0 = 0.4 \cdot 1 + (-4.9) \cdot 5 + (-200.3) \cdot 150 = 0.4 - 24.5 - 30045 = -30069.1$$

Category 1:

$$\text{score}_1 = 0.3 \cdot 1 + 4.8 \cdot 5 + 200.5 \cdot 150 = 0.3 + 24 + 30075 = 30099.3$$

Category 2:

$$\text{score}_2 = (-0.1) \cdot 1 + 0.3 \cdot 5 + 0.2 \cdot 150 = -0.1 + 1.5 + 30 = 31.4$$

Step 3: Predict category

$$\hat{y}^{(3)} = \arg\max(-30069.1, 30099.3, 31.4) = 1$$

Step 4: Check if correct
True label: $y^{(3)} = 2$, Predicted: $\hat{y}^{(3)} = 1$
Prediction is WRONG! → Update weights
Step 5: Update weights
Increase row 2 (true category):

$$\mathbf{w}_2^{\text{new}} = [-0.1, 0.3, 0.2] + [1, 5, 150] = [0.9, 5.3, 150.2]$$

Decrease row 1 (predicted category):

$$\mathbf{w}_1^{\text{new}} = [0.3, 4.8, 200.5] - [1, 5, 150] = [-0.7, -0.2, 50.5]$$

Row 0 unchanged:

$$\mathbf{w}_0^{\text{new}} = [0.4, -4.9, -200.3]$$

Final weight matrix after all samples:

$$W = \begin{bmatrix} 0.4 & -4.9 & -200.3 \\ -0.7 & -0.2 & 50.5 \\ 0.9 & 5.3 & 150.2 \end{bmatrix}$$

(b) Predict for new user: Excitement = 6, Budget = 200

Step 1: Form input vector

$$\mathbf{x}^{\text{new}} = \begin{bmatrix} 1 \\ 6 \\ 200 \end{bmatrix}$$

Step 2: Calculate scores using final weights
Category 0:

$$\text{score}_0 = 0.4 \cdot 1 + (-4.9) \cdot 6 + (-200.3) \cdot 200 = 0.4 - 29.4 - 40060 = -40089$$

Category 1:

$$\text{score}_1 = (-0.7) \cdot 1 + (-0.2) \cdot 6 + 50.5 \cdot 200 = -0.7 - 1.2 + 10100 = 10098.1$$

Category 2:

$$\text{score}_2 = 0.9 \cdot 1 + 5.3 \cdot 6 + 150.2 \cdot 200 = 0.9 + 31.8 + 30040 = 30072.7$$

Step 3: Predict category

$$\hat{y}^{\text{new}} = \arg\max(-40089, 10098.1, 30072.7) = 2$$

Predicted Activity Category: 2 (Relaxation spots)