Linear regression can be used to quantify the effect of advertisement on sales. Various channels of advertisement are newspaper, TV, radio, etc.
I will have a look at the advertising data of a company and try to see its effect on sales.

Data Information

The data at hand has three features about the spending on advertising and the target variable is the net sales. Attributes are:

- TV - Independent variable quantifying budget for TV ads
- Radio - Independent variable quantifying budget for radio ads
- News - Independent variable quantifying budget for news ads
- Sales - Dependent variable

```
import pandas as pd
import numpy as np
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```

1. Import the dataset.

```
data = pd.read_csv('sample_data/Advertising.csv')
Ad_df = data.copy()
```

2. Check the dataset.

```
Ad_df.head()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

```
print(f"There are {Ad_df.shape[0]} rows and {Ad_df.shape[1]} columns.")
```

```
There are 200 rows and 5 columns.
```

```
Ad_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  200 non-null    int64  
 1   TV          200 non-null    float64
 2   Radio       200 non-null    float64
 3   Newspaper   200 non-null    float64
 4   Sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

```
Ad_df.describe(include='all').T
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	200.0	100.5000	57.879185	1.0	50.750	100.50	150.250	200.0
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
Newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
Sales	200.0	14.0225	5.217457	1.6	10.375	12.90	17.400	27.0

3. Check missing and duplicated values.

```
Ad_df.isnull().sum()
```

	0
Unnamed: 0	0
TV	0
Radio	0
Newspaper	0
Sales	0

dtype: int64

```
Ad_df.duplicated().sum()
```

```
0
```

4. Drop column *Unnamed* as it is just the index.

```
#Your answer

Ad_df.drop(columns=['Unnamed: 0'], inplace=True)
```

** Now I will omplement a linear regression function using numpy only and name it lin_reg. It should take in the independent variable(s) and the observation vector. It should return the prediction.**

```
#Your answer
#Implement a linear regression function using numpy only.

def lin_reg(X, y):
    X = np.column_stack([np.ones(X.shape[0],dtype = 'int'), X])

    # Calculate the coefficients (beta)
    beta_hat = np.linalg.inv(X.T @ X) @ X.T @ y

    # Make predictions
    y_pred = X @ beta_hat

    return y_pred
```

First I will start with the simple linear regression. I will use lin_reg with one independent variable at a time.

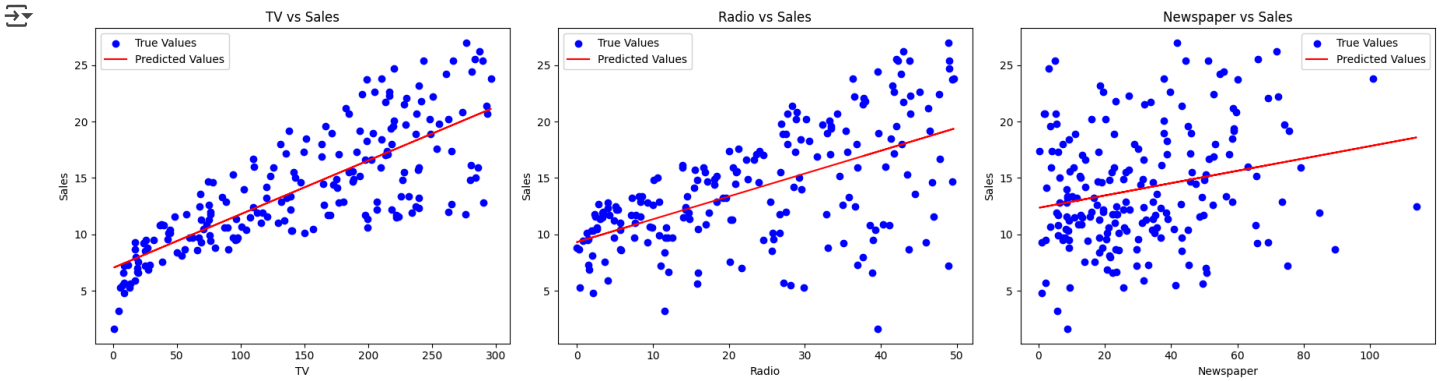
```
Sales = Ad_df.Sales.values.reshape(len(Ad_df['Sales']), 1)
TV = Ad_df.TV.values.reshape(len(Ad_df['Sales']), 1)
Radio = Ad_df.Radio.values.reshape(len(Ad_df['Sales']), 1)
Newspaper = Ad_df.Newspaper.values.reshape(len(Ad_df['Sales']), 1)
```

```
#prediction for TV vs Sales
X_tv = TV
```

```
y = Sales
y_pred_TV = lin_reg(X_tv, y)


#prediction for Radio vs Sales
X_Radio = Radio
y_pred_Radio = lin_reg(X_Radio, y)

#prediction of Newspaper vs Sales

X_Newspaper = Newspaper
y_pred_Newspaper = lin_reg(X_Newspaper, y)
```

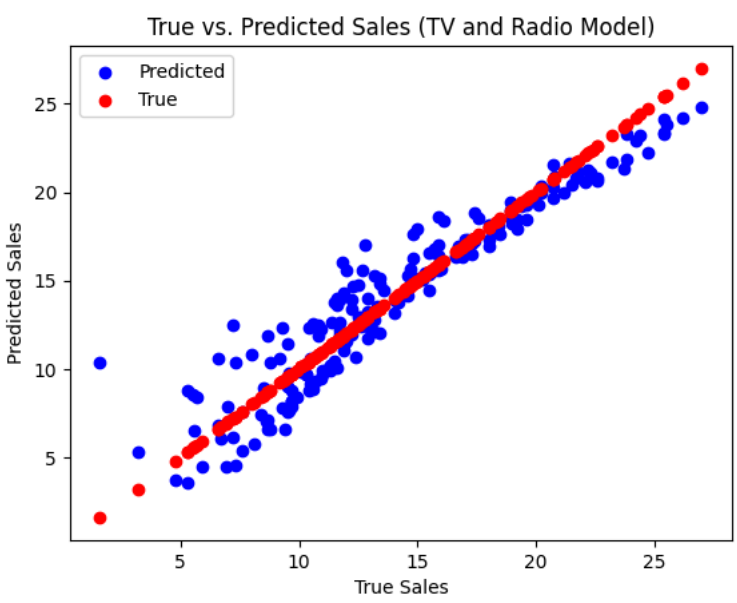
For each created model I plotted the predicted values against the true values.

```
# Your answer

import matplotlib.pyplot as plt

plt.figure(figsize=(18, 5))

# TV vs Sales
plt.subplot(1, 3, 1)
plt.scatter(X_tv, y, color='blue', label='True Values')
plt.plot(X_tv, y_pred_TV, color='red', label='Predicted Values')
plt.title('TV vs Sales')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.legend()

# Radio vs Sales
plt.subplot(1, 3, 2)
plt.scatter(X_Radio, y, color='blue', label='True Values')
plt.plot(X_Radio, y_pred_Radio, color='red', label='Predicted Values')
plt.title('Radio vs Sales')
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.legend()

# Newspaper vs Sales
plt.subplot(1, 3, 3)
plt.scatter(X_Newspaper, y, color='blue', label='True Values')
plt.plot(X_Newspaper, y_pred_Newspaper, color='red', label='Predicted Values')
plt.title('Newspaper vs Sales')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.legend()

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



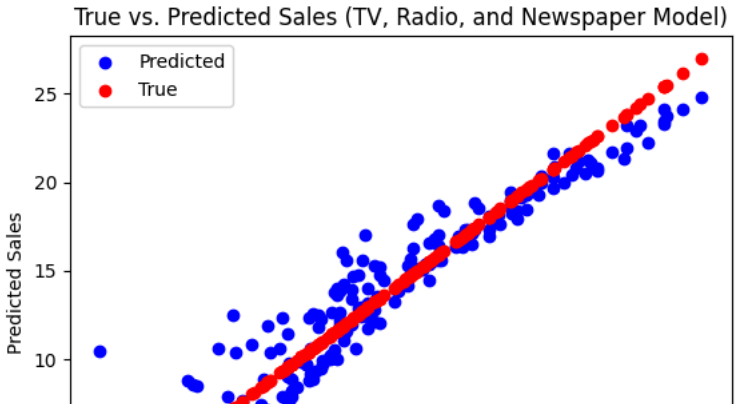
I will not create a multiple linear regression model with TV and Radio. I will plot the predicted values against the true values.

```
#Your answer

import matplotlib.pyplot as plt

X_TV_Radio = Ad_df[['TV', 'Radio']].values
y = Sales
predictions_TV_Radio = lin_reg(X_TV_Radio, y)

# Scatter plot of true values vs. predicted values
plt.scatter(y, predictions_TV_Radio, color='blue', label='Predicted')  # Predicted values
plt.scatter(y, y, color='red', label='True')  # True values (45-degree line)


plt.xlabel("True Sales")
plt.ylabel("Predicted Sales")
plt.title("True vs. Predicted Sales (TV and Radio Model)")
plt.legend()
plt.show()
```



Next, I will create a multiple linear regression model with all three independent variables. I will plot the predicted values against the true values.

```
#Your answer

import matplotlib.pyplot as plt

X_TV_Radio = Ad_df[['TV', 'Radio', 'Newspaper']].values
y = Sales
predictions_TV_Radio_Newspaper = lin_reg(X_TV_Radio, y)

# Scatter plot of true values vs. predicted values
plt.scatter(y, predictions_TV_Radio_Newspaper, color='blue', label='Predicted')  # Predicted values
plt.scatter(y, y, color='red', label='True')  # True values (45-degree line)

plt.xlabel("True Sales")
plt.ylabel("Predicted Sales")
plt.title("True vs. Predicted Sales (TV, Radio, and Newspaper Model)")
plt.legend()
plt.show()
```



For each of the five models, I will compute the MSE (mean squared error), which is defined as

$$\frac{\sum_{i=1}^{n} (\hat{y_i} - y_i)^2}{n}$$

That is, the mean of the squared differences between the *predicted* values $\hat{y}_i = x_i^T \beta$ and the *true* values y_i. Note that x_i^T is just the i-th row of the design matrix, corresponding to the i-th observation.

```
import numpy as np

# Assuming you have the predictions for each model (y_pred_TV, y_pred_Radio, etc.) and the true values (Sales)

# Calculate MSE for each model
mse_TV = np.mean((y_pred_TV - Sales)**2)
mse_Radio = np.mean((y_pred_Radio- Sales)**2)
mse_Newspaper = np.mean((y_pred_Newspaper - Sales)**2)
mse_TV_Radio = np.mean((predictions_TV_Radio - Sales)**2)
mse_TV_Radio_Newspaper = np.mean((predictions_TV_Radio_Newspaper - Sales)**2)


# Print the MSE values
print("MSE (TV Model):", mse_TV)
print("MSE (Radio Model):", mse_Radio)
print("MSE (Newspaper Model):", mse_Newspaper)
print("MSE (TV and Radio Model):", mse_TV_Radio)
print("MSE (TV, Radio Model, and Newspaper Model):", mse_TV_Radio_Newspaper)
```

```
MSE (TV Model): 10.512652915656757
MSE (Radio Model): 18.09239774512544
MSE (Newspaper Model): 25.674022720559698
MSE (TV and Radio Model): 2.784569900338091
MSE (TV, Radio Model, and Newspaper Model): 2.784126314510936
```