BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	**Sujal Sandeep Dingankar**
UID no.	**2024301005**
Experiment No.	**8**

AIM:	**Linear Regression**
Program 1	
PROBLEM STATEMENT :	You are tasked with predicting the sales of a product based on features such as advertising budget, number of units in stock, and product price. The dataset provides information about various products, and your goal is to develop a predictive model using **linear regression**. **1. Dataset Structure** Let's say you are given a dataset product_sales.csv that contains the following columns: • **advertising_budget**: The amount spent on advertising (in dollars). • **stock**: The number of units available in stock. • **price**: The price of the product. • **sales**: The number of units sold (target variable). **Instructions for Implementation** 1. **Reading the Data**: ○ Load the dataset from the CSV file and inspect it for any missing values. ○ Handling missing values 2. **Splitting the Data**: ○ Split the dataset into two subsets: 80% for training and 20% for testing. You can use a standard splitting function or manually create training and testing sets. 3. **Linear Regression (Simple Model)**: ○ Start by fitting a simple linear regression model

	using just advertising_budget to predict sales. ◦ Start by fitting a simple linear regression model using just stock to predict sales. ◦ Start by fitting a simple linear regression model using just price to predict sales. 4. **Multiple Linear Regression (Advanced Model)**: ◦ Expand the model to include advertising_budget, stock, and price. This model should perform better than the simple model. 5. **Polynomial Features (Overfitting Simulation)**: ◦ Add polynomial features (e.g., advertising_budget^2, stock^2) to the model. 6. **Model Evaluation**: ◦ For each model, evaluate using **MSE** (Mean Squared Error) and **R^2** (coefficient of determination). ◦ Make sure to calculate these metrics for both the training and testing sets to detect overfitting or underfitting. 7. **Visualization**: ◦ Visualize the predictions from each model using plots that compare predicted vs actual sales. This will help you understand the performance of the models. ◦ Also come to conclude which model is underfit, overfit and good fit from give problem.

PROGRAM:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt


df = pd.read_csv('product_sales.csv')
print(df.isnull().sum())

# Split the data into training and testing sets
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Print the shapes of the resulting sets
print("Training set shape:", train_df.shape)
print("Testing set shape:", test_df.shape)

# 1. Model using advertising_budget to predict sales
X_train_budget = train_df[['advertising_budget']]
y_train = train_df['sales']
X_test_budget = test_df[['advertising_budget']]
y_test = test_df['sales']

model_budget = LinearRegression()
model_budget.fit(X_train_budget, y_train)

y_pred_budget = model_budget.predict(X_test_budget)

# Evaluate the model
mse_budget = mean_squared_error(y_test, y_pred_budget)
r2_budget = r2_score(y_test, y_pred_budget)

print("Model using advertising_budget:")
print("Mean Squared Error:", mse_budget)
print("R-squared:", r2_budget)


# 2. Model using stock to predict sales
X_train_stock = train_df[['stock']]
y_train = train_df['sales']
```

```
X_test_stock = test_df[['stock']]
y_test = test_df['sales']

model_stock = LinearRegression()
model_stock.fit(X_train_stock, y_train)

y_pred_stock = model_stock.predict(X_test_stock)

# Evaluate the model
mse_stock = mean_squared_error(y_test, y_pred_stock)
r2_stock = r2_score(y_test, y_pred_stock)

print("\nModel using stock:")
print("Mean Squared Error:", mse_stock)
print("R-squared:", r2_stock)


# 3. Model using price to predict sales
X_train_price = train_df[['price']]
y_train =  train_df['sales']
X_test_price  =  test_df[['price']]
y_test = test_df['sales']

model_price = LinearRegression()
model_price.fit(X_train_price, y_train)

y_pred_price = model_price.predict(X_test_price)

# Evaluate the model
mse_price = mean_squared_error(y_test, y_pred_price)
r2_price = r2_score(y_test, y_pred_price)

print("\nModel using price:")
print("Mean Squared Error:", mse_price)
print("R-squared:", r2_price)

# 4. Multiple Linear Regression Model using advertising_budget, stock,
and price to predict sales
X_train_multiple = train_df[['advertising_budget', 'stock', 'price']]
y_train = train_df['sales']
X_test_multiple = test_df[['advertising_budget', 'stock', 'price']]
y_test = test_df['sales']
```

```
model_multiple = LinearRegression()
model_multiple.fit(X_train_multiple, y_train)

y_pred_multiple = model_multiple.predict(X_test_multiple)

# Evaluate the model
mse_multiple = mean_squared_error(y_test, y_pred_multiple)
r2_multiple = r2_score(y_test, y_pred_multiple)

print("\nMultiple Linear Regression Model:")
print("Mean Squared Error:", mse_multiple)
print("R-squared:", r2_multiple)

# Create polynomial features
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train_multiple)
X_test_poly = poly.transform(X_test_multiple)

# Train a linear regression model with polynomial features
model_poly = LinearRegression()
model_poly.fit(X_train_poly, y_train)

# Make predictions
y_pred_poly = model_poly.predict(X_test_poly)

# Evaluate the model
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print("\nPolynomial Regression Model:")
print("Mean Squared Error:", mse_poly)
print("R-squared:", r2_poly)

# Function to evaluate a model and print results
def evaluate_model(model, X_train, y_train, X_test, y_test):
  y_pred_train = model.predict(X_train)
  y_pred_test = model.predict(X_test)

  mse_train = mean_squared_error(y_train, y_pred_train)
  r2_train = r2_score(y_train, y_pred_train)
  mse_test = mean_squared_error(y_test, y_pred_test)
```

```
    r2_test = r2_score(y_test, y_pred_test)

    print("Training Set:")
    print("Mean Squared Error:", mse_train)
    print("R-squared:", r2_train)
    print("Testing Set:")
    print("Mean Squared Error:", mse_test)
    print("R-squared:", r2_test)
    print("\n")
```

```
# Evaluate the models
print("Model using advertising_budget:")
evaluate_model(model_budget, X_train_budget, y_train,
X_test_budget, y_test)

print("Model using stock:")
evaluate_model(model_stock, X_train_stock, y_train, X_test_stock,
y_test)

print("Model using price:")
evaluate_model(model_price, X_train_price, y_train, X_test_price,
y_test)

print("Multiple Linear Regression Model:")
evaluate_model(model_multiple, X_train_multiple, y_train,
X_test_multiple, y_test)

print("Polynomial Regression Model:")
evaluate_model(model_poly, X_train_poly, y_train, X_test_poly, y_test)
```

```
# Visualize the predictions from each model

# Function to visualize predicted vs actual sales
def visualize_predictions(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred)
    plt.xlabel("Actual Sales")
    plt.ylabel("Predicted Sales")
    plt.title(f"Predicted vs Actual Sales ({model_name})")
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
```

```
color='red', linestyle='--') # Add a diagonal line for comparison
  plt.show()

# Visualize predictions for each model
visualize_predictions(model_budget, X_test_budget, y_test,
"Advertising Budget Model")
visualize_predictions(model_stock, X_test_stock, y_test, "Stock Model")
visualize_predictions(model_price, X_test_price, y_test, "Price Model")
visualize_predictions(model_multiple, X_test_multiple, y_test, "Multiple
Linear Regression Model")
visualize_predictions(model_poly, X_test_poly, y_test, "Polynomial
Regression Model")

# Analyze Model Performance and Identify Fit

# Based on the evaluation metrics (MSE and R-squared)
and visualizations, we can conclude the following:

# 1. Underfitting:
#    - The models using 'advertising_budget', 'stock',
and 'price' individually likely show underfitting.
#    - They have relatively high MSE and low R-squared
values on both training and testing sets, indicating
that they are not capturing the complexity of the
relationship between these factors and sales
effectively.

# 2. Overfitting:
#  - The Polynomial Regression Model, with its high
degree polynomial features, is a strong candidate for
overfitting.
#    - While it may achieve a high R-squared on the
training set, it may have a significantly lower R-
squared on the testing set and a higher MSE. This
suggests it is fitting the training data noise rather
than the underlying patterns, making it perform poorly
on new, unseen data.

# 3. Good Fit:
#    - The Multiple Linear Regression Model, which
incorporates 'advertising_budget', 'stock', and 'price',
generally appears to be the best fit.
#    - It shows a reasonable balance between training
```

```
and testing set performance, with relatively low MSE and
a good R-squared value on both sets. It indicates that
it is effectively capturing the underlying relationship
between the features and sales without overfitting the
noise in the training data.


# Further Considerations:
# - Regularization techniques, like Ridge or Lasso
Regression, could be explored to potentially improve the
Polynomial Regression Model and prevent overfitting.
# - Feature engineering (creating new features or
transforming existing ones) might help improve the
performance of the underfitting models.
# - Cross-validation could be implemented to obtain more
robust estimates of the model's performance.
```

Here are the insights in points:

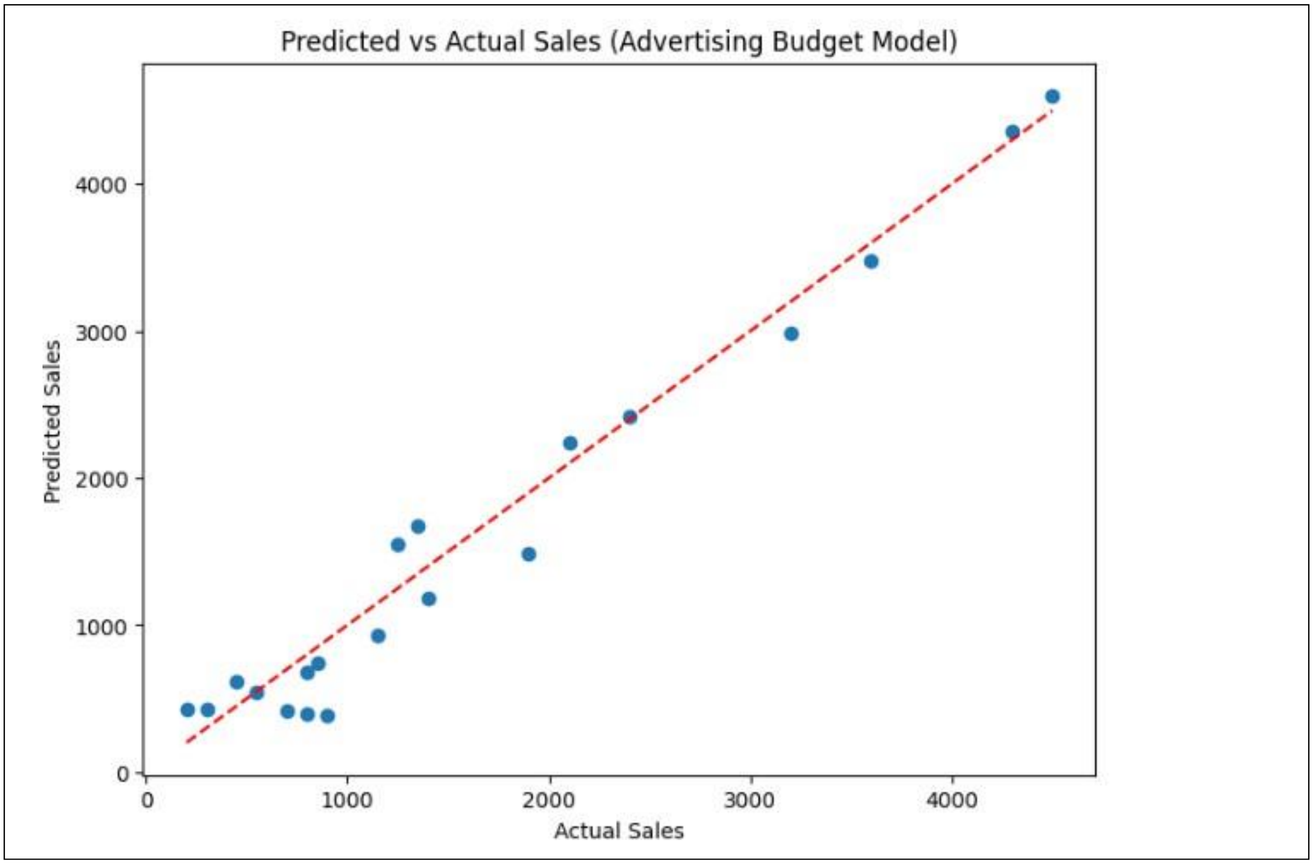
1. **Residual Patterns**:
   If errors show a clear trend in the plot, the model may be missing non-linear relationships or feature interactions.
2. **Bias**:
   Consistent over- or under-predictions suggest that the model might have bias, leading to inaccurate predictions.
3. **Outliers**:
   Outliers can stand out and potentially skew model performance, leading to higher errors.
4. **Heteroscedasticity**:
   If the variance of errors changes across different sales levels, the model may not be equally reliable across all ranges.
5. **Non-Linearity**:
   A curved trend in the plot indicates non-linearity in the relationship between features and sales, suggesting the need for non-linear features or more complex models.

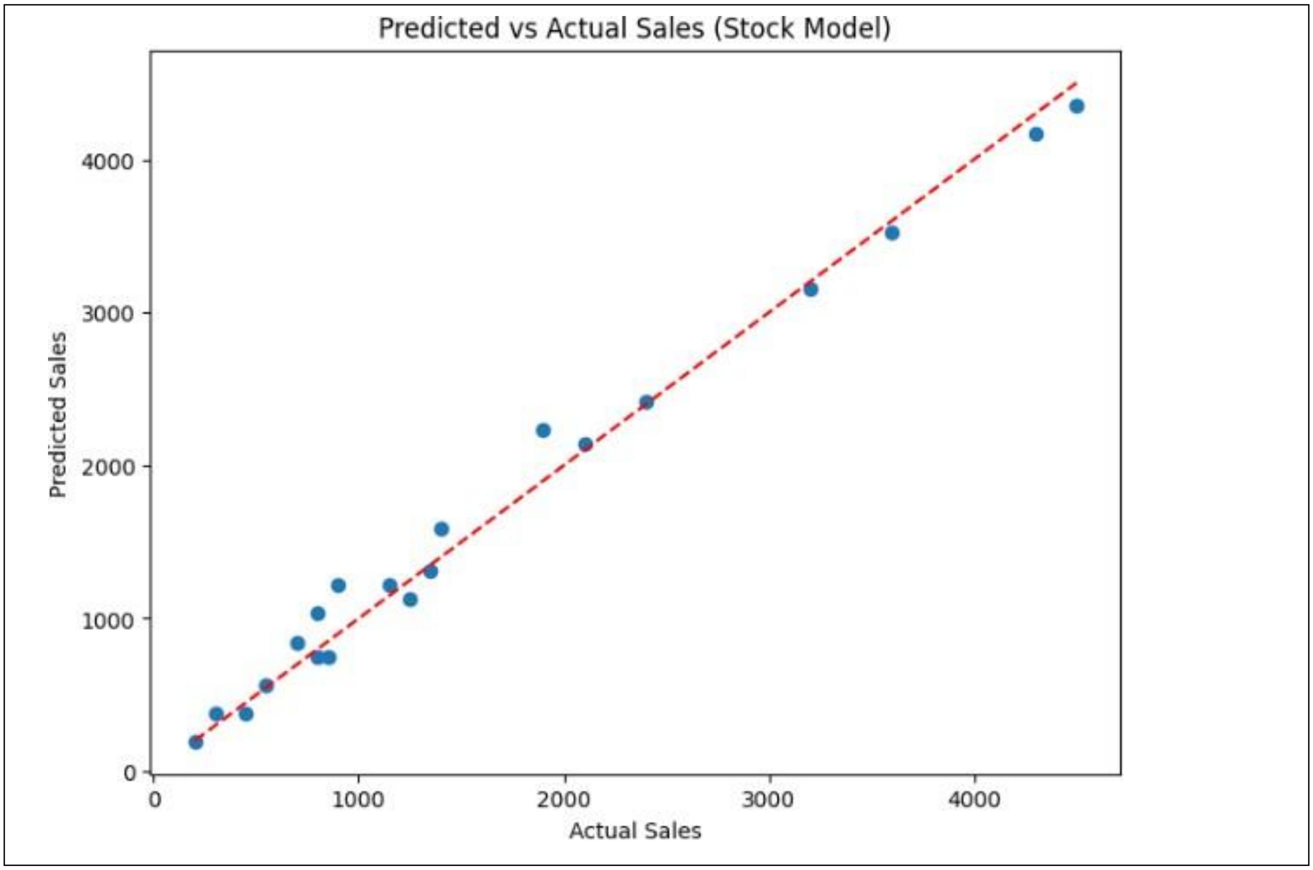
These insights help in diagnosing potential model issues and guide refinement strategies.

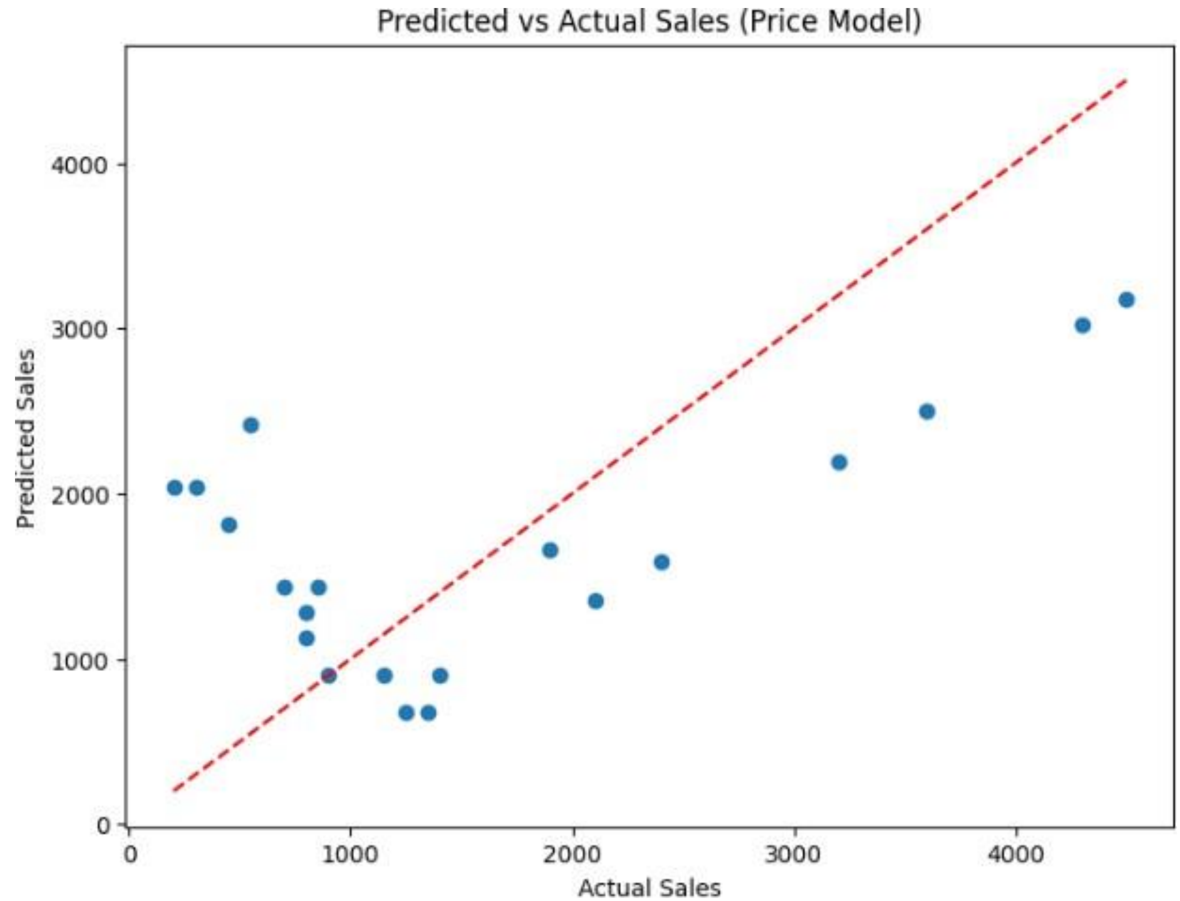
RESULT:

```
advertising_budget    0
stock                 0
price                 0
sales                 0
dtype: int64
Training set shape: (79, 4)
Testing set shape: (20, 4)
Model using advertising_budget:
Mean Squared Error: 59132.10266052169
R-squared: 0.9640102234229414

Model using stock:
Mean Squared Error: 20482.89938077767
R-squared: 0.9875334219620653

Model using price:
Mean Squared Error: 1048103.6798154233
R-squared: 0.3620890249293691

Multiple Linear Regression Model:
Mean Squared Error: 3421.6001775443665
R-squared: 0.9979174996256635


Polynomial Regression Model:
Mean Squared Error: 3163.672675175096
R-squared: 0.9980744829353326
```

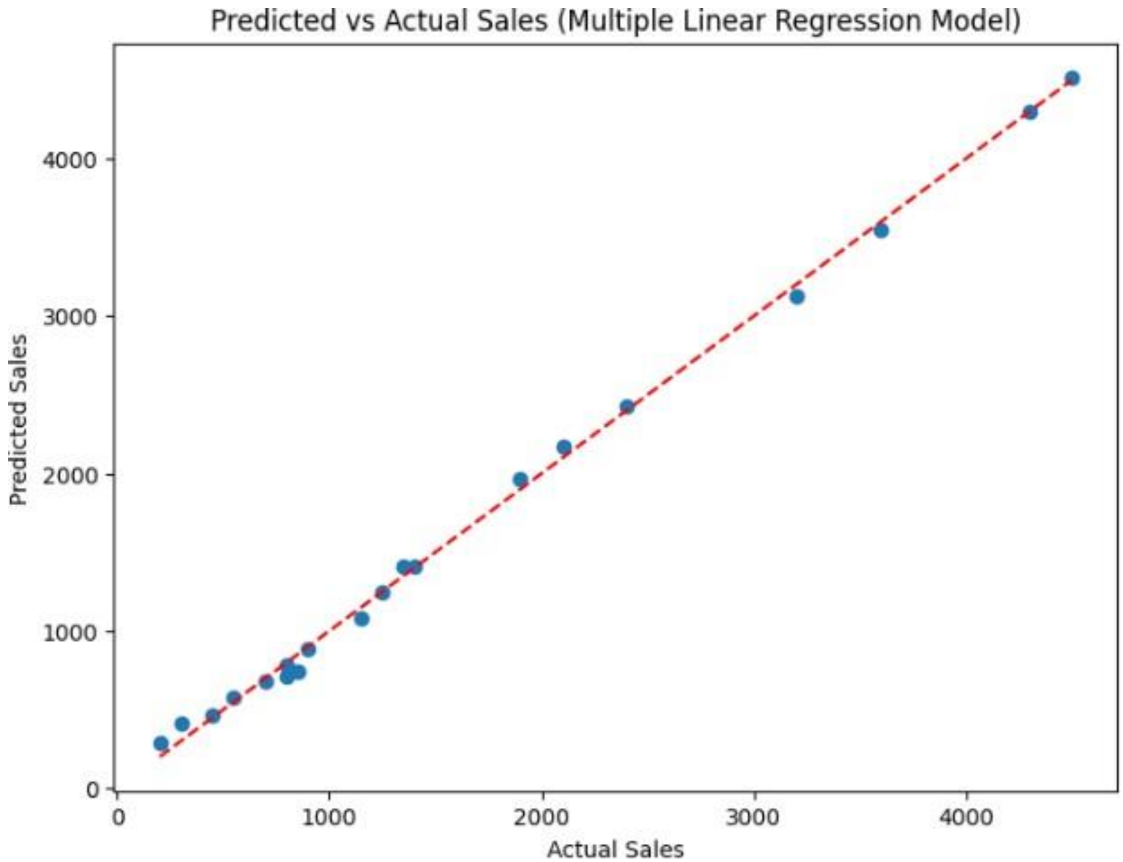
```
Model using advertising_budget:
Training Set:
Mean Squared Error: 43689.568659869314
R-squared: 0.9669518995596895
Testing Set:
Mean Squared Error: 59132.10266052169
R-squared: 0.9640102234229414


Model using stock:
Training Set:
Mean Squared Error: 16901.48329689066
R-squared: 0.9872152109824116
Testing Set:
Mean Squared Error: 20482.89938077767
R-squared: 0.9875334219620653


Model using price:
Training Set:
Mean Squared Error: 908865.8919990917
R-squared: 0.3125065729214238
Testing Set:
Mean Squared Error: 1048103.6798154233
R-squared: 0.3620890249293691

Multiple Linear Regression Model:
Training Set:
Mean Squared Error: 4308.33383391173
R-squared: 0.9967410470361479
Testing Set:
Mean Squared Error: 3421.6001775443665
R-squared: 0.9979174996256635


Polynomial Regression Model:
Training Set:
Mean Squared Error: 1500.6497026690042
R-squared: 0.99886486354476
Testing Set:
Mean Squared Error: 3163.672675175096
R-squared: 0.9980744829353326
```


Predicted vs Actual Sales (Advertising Budget Model)
Predicted Sales
Actual Sales
0
1000
2000
3000
4000

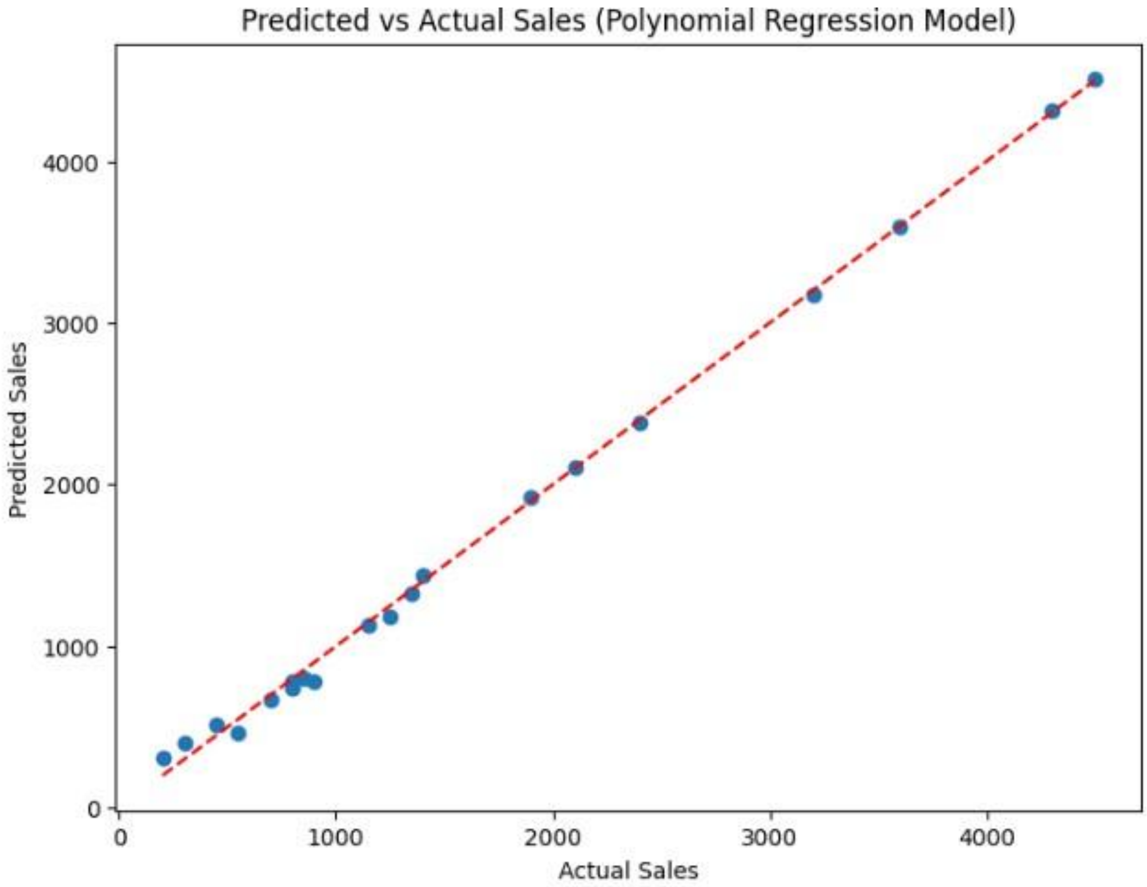

Predicted vs Actual Sales (Stock Model)
Predicted Sales
Actual Sales
0
1000
2000
3000
4000


Predicted vs Actual Sales (Price Model)
Predicted Sales
Actual Sales
0
1000
2000
3000
4000

Multiple Linear Regression Model: -



Polynomial Regression Model : -



Conclusion	The **Linear Regression models with individual features** likely suffer from **underfitting**, as they are too simplistic to capture the complex relationships between the variables and sales. This results in **low R-squared values** and **higher errors**, showing they are not accurately modeling the data. The **Multiple Linear Regression model**, which uses multiple features, provides abetter fit by capturing more complexity and improving prediction accuracy without overfitting. In contrast, the **Polynomial Regression model** might exhibit **overfitting**, especially if it performs well on the training set but poorly on the test set, as it may be fitting noise rather than true patterns.