EXP NO: 3	Model Planning and Building
DATE: 29/7/25	

Aim:

To implement and compare machine learning models (Linear, Polynomial, and SVR) and apply KMeans clustering for data segmentation.

Program:

Step 1: Import Required Libraries

```
import   pandas   as   pd   import   numpy   as   np   import   seaborn   as   sns   import
matplotlib.pyplot as plt from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR from sklearn.pipeline
import    Pipeline    from    sklearn.metrics    import    mean_absolute_error,
mean_squared_error
```

from sklearn.cluster import KMeans **Step 2: Load and Explore the Dataset**

```
df = pd.read_csv('Advertising.csv')
```

```
print("\nData Info:") df.info()
```

Step 3: Define Features and Target

```
X = df[['TV', 'Radio', 'Newspaper']]
= df['Sales']                          y
```

Step 4: Visualize Data Relationships

```
sns.pairplot(df, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=4,
aspect=1, kind='reg') plt.show()
```

Step 5: Split Data into Training and Testing Sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 6: Train Linear Regression Model

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train) y_pred_lr

= lr_model.predict(X_test)
```

Step 7: Train Polynomial Regression Model

```
poly_pipeline = Pipeline([

    ('poly_features', PolynomialFeatures(degree=2, include_bias=False)),
('linear_regression', LinearRegression())
])
poly_pipeline.fit(X_train, y_train) y_pred_poly
=  poly_pipeline.predict(X_test)
```

Step 8: Train Support Vector Regression (SVR) Model

```
svr_model = SVR(kernel='rbf') svr_model.fit(X_train, y_train) y_pred_svr =
svr_model.predict(X_test)
```

Step 9: Evaluate All Models

```
models = {


    'Linear Regression': y_pred_lr,
    'Polynomial Regression': y_pred_poly,
    'Support Vector Regression': y_pred_svr
}
evaluation_results = {}
for name, y_pred in models.items():      mae
= mean_absolute_error(y_test, y_pred)     mse
= mean_squared_error(y_test, y_pred)     rmse
= np.sqrt(mse)
    evaluation_results[name] = {'MAE': mae, 'MSE': mse, 'RMSE': rmse}
print(f"\n---{name}---")     print(f"MAE:{mae:.4f}")     print(f"MSE:
{mse:.4f}")        print(f"RMSE:{rmse:.4f}")
```

Step10:Visualize Model Comparison

```
results_df = pd.DataFrame(evaluation_results).T
results_df['RMSE'].plot(kind='bar', figsize=(10, 6))
plt.title('Model Comparison by RMSE')
plt.ylabel('Root Mean Squared Error')
plt.xticks(rotation=0) plt.show()
```

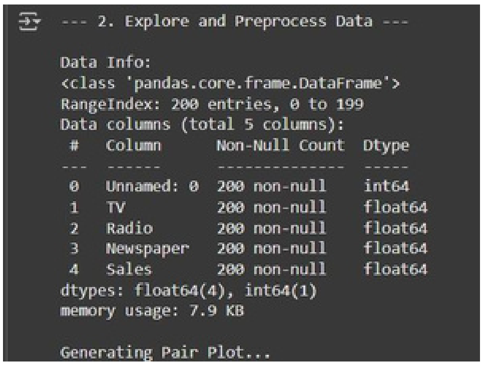
Step 11: Apply K-Means Clustering

```
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X)
```

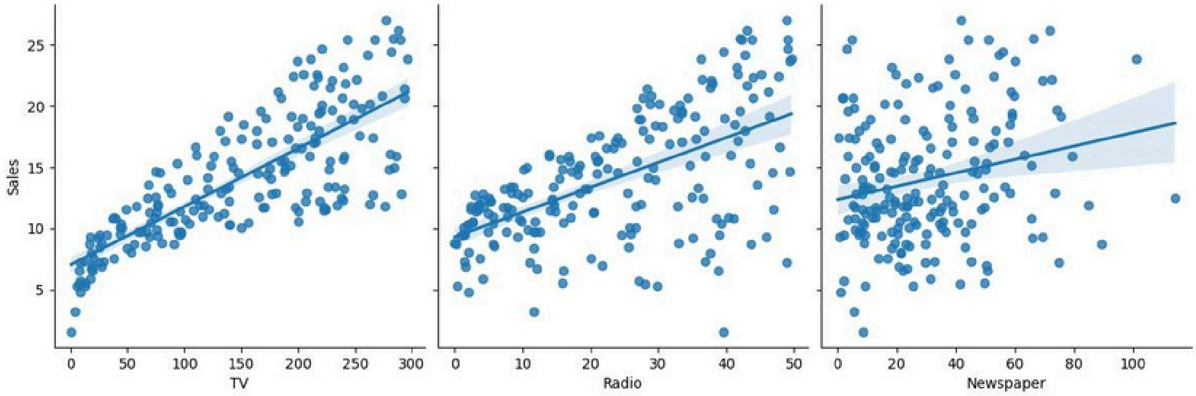
Step 12: Visualize Clusters

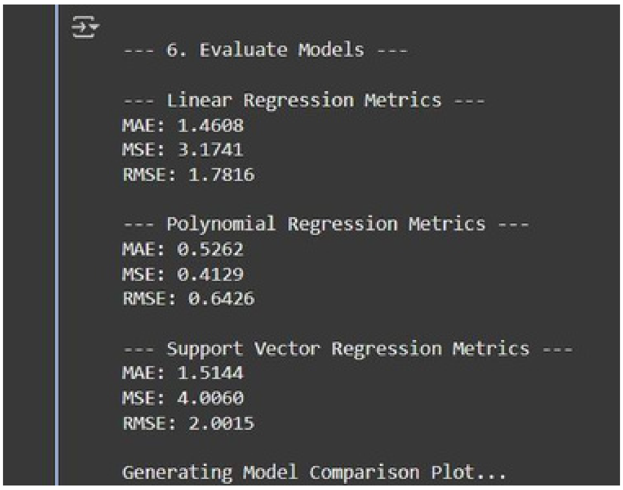
```
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df, x='TV', y='Sales', hue='Cluster', palette='viridis',
s=100, alpha=0.7)
plt.title('TV vs Sales by Cluster') plt.show()
```

Output:

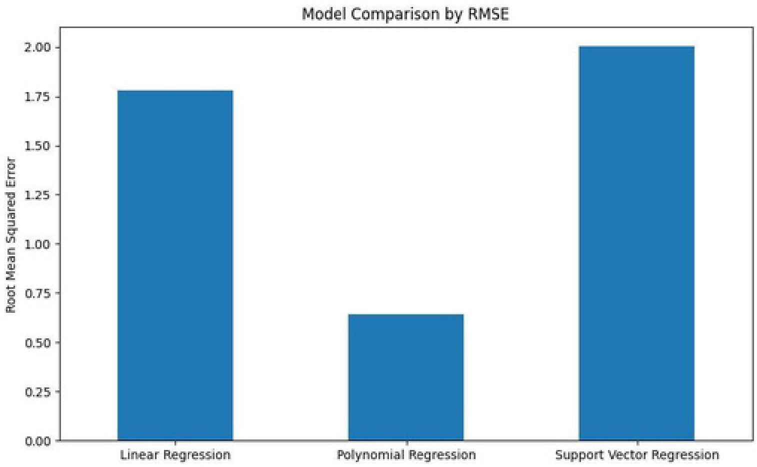


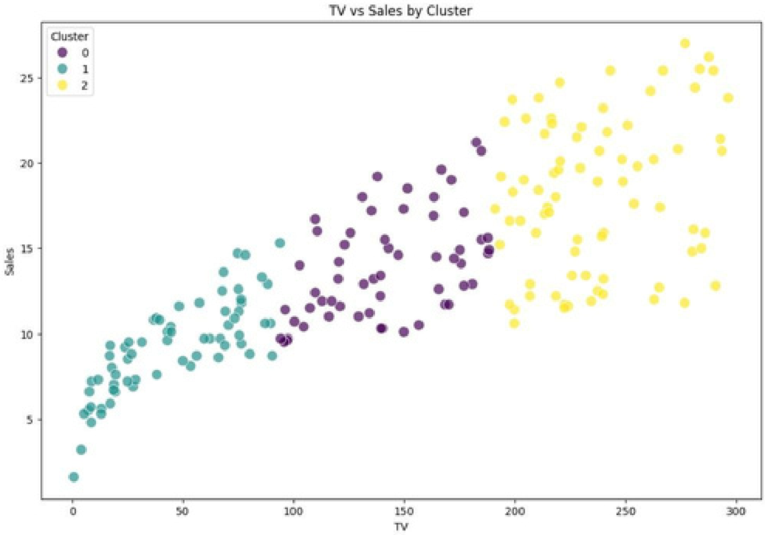
```
--- 2. Explore and Preprocess Data ---

Data 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

Generating Pair Plot...
```





```
--- 6. Evaluate Models ---

--- Linear Regression Metrics ---
MAE: 1.4608
MSE: 3.1741
RMSE: 1.7816

--- Polynomial Regression Metrics ---
MAE: 0.5262
MSE: 0.4129
RMSE: 0.6426

--- Support Vector Regression Metrics ---
MAE: 1.5144
MSE: 4.0060
RMSE: 2.0015

Generating Model Comparison Plot...
```





Result:

The models were successfully trained, evaluated, and visualized. K-Means clustering grouped the data into three meaningful clusters based on sales and advertising features.