Assignment 6

Exercises 6 and 10

Exercise 6(a)

Could not get wage data to load to complete exercise 6

10(a)

In [36]:
```
import os
os.getcwd()
```

Out[36]:
```
'/Users/shamircardenas/Documents/STA-6543-9IT-Summer 2025-Predictive
Modeling/Assignment 6'
```

In [46]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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 StandardScaler
import warnings
warnings.filterwarnings('ignore')

College = pd.read_csv('/Users/shamircardenas/Documents/STA-6543-9IT-Su
```

In [48]:
```
print("Dataset shape:", College.shape)
print("\nColumn names:")
print(College.columns.tolist())
print("\nFirst few rows:")
print(College.head())

print("\nMissing values:")
print(College.isnull().sum())
```

```
Dataset shape: (777, 19)

Column names:
['Unnamed: 0', 'Private', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top
25perc', 'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board', 'Book
s', 'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expen
d', 'Grad.Rate']
```

First few rows:

	Unnamed: 0	Private	Apps	Accept	Enroll	Top10perc
0	Abilene Christian University	Yes	1660	1232	721	23
1	Adelphi University	Yes	2186	1924	512	16
2	Adrian College	Yes	1428	1097	336	22
3	Agnes Scott College	Yes	417	349	137	60
4	Alaska Pacific University	Yes	193	146	55	16

	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal
0	52	2885	537	7440	3300	450	2200
1	29	2683	1227	12280	6450	750	1500
2	50	1036	99	11250	3750	400	1165
3	89	510	63	12960	5450	450	875
4	44	249	869	7560	4120	800	1500

	PhD	Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
0	70	78	18.1	12	7041	60
1	29	30	12.2	16	10527	56
2	53	66	12.9	30	8735	54
3	92	97	7.7	37	19016	59
4	76	72	11.9	2	10922	15

Missing values:

Unnamed: 0	0
Private	0
Apps	0
Accept	0
Enroll	0
Top10perc	0
Top25perc	0
F.Undergrad	0
P.Undergrad	0
Outstate	0
Room.Board	0
Books	0
Personal	0
PhD	0
Terminal	0
S.F.Ratio	0
perc.alumni	0

```
Expend         0
Grad.Rate      0
dtype: int64
```

In [55]:

```
target_column = 'Outstate'

numeric_columns = College.select_dtypes(include=[np.number]).columns.t
if target_column in numeric_columns:
    feature_columns = [col for col in numeric_columns if col != target
else:
    print(f"Warning: {target_column} not found in numeric columns")
    print("Available numeric columns:", numeric_columns)


X = College[feature_columns]
y = College[target_column]

print(f"\nTarget variable: {target_column}")
print(f"Number of features: {len(feature_columns)}")
print("Features:", feature_columns)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

print(f"\nTraining set size: {X_train.shape[0]} samples")
print(f"Test set size: {X_test.shape[0]} samples")


def forward_stepwise_selection(X_train, y_train, X_test, y_test, max_f
    """
    Perform forward stepwise selection
    """
    if max_features is None:
        max_features = X_train.shape[1]

    selected_features = []
    remaining_features = list(X_train.columns)

    results = []

    print("Starting Forward Stepwise Selection...")
    print("=" * 50)

    for step in range(min(max_features, len(remaining_features))):
        best_score = -np.inf
        best_feature = None


        for feature in remaining_features:

            temp_features = selected_features + [feature]
```

```
            model = LinearRegression()
            model.fit(X_train[temp_features], y_train)

            train_pred = model.predict(X_train[temp_features])
            train_r2 = r2_score(y_train, train_pred)

            if train_r2 > best_score:
                best_score = train_r2
                best_feature = feature

        if best_feature is not None:
            selected_features.append(best_feature)
            remaining_features.remove(best_feature)

            model = LinearRegression()
            model.fit(X_train[selected_features], y_train)

            train_pred = model.predict(X_train[selected_features])
            train_r2 = r2_score(y_train, train_pred)
            train_mse = mean_squared_error(y_train, train_pred)

            test_pred = model.predict(X_test[selected_features])
            test_r2 = r2_score(y_test, test_pred)
            test_mse = mean_squared_error(y_test, test_pred)

            results.append({
                'step': step + 1,
                'feature_added': best_feature,
                'features': selected_features.copy(),
                'n_features': len(selected_features),
                'train_r2': train_r2,
                'train_mse': train_mse,
                'test_r2': test_r2,
                'test_mse': test_mse
            })

            print(f"Step {step + 1}: Added '{best_feature}'")
            print(f"  Features: {selected_features}")
            print(f"  Train R²: {train_r2:.4f}, Test R²: {test_r2:.4f}
            print(f"  Train MSE: {train_mse:.2f}, Test MSE: {test_mse:
            print()

    return results, selected_features
```

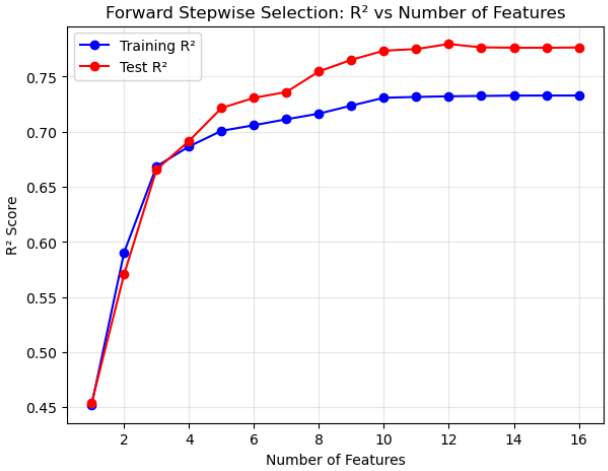
```
stepwise_results, final_features = forward_stepwise_selection(X_train,


results_df = pd.DataFrame(stepwise_results)
print("Forward Stepwise Selection Results:")
print("=" * 50)
print(results_df[['step', 'feature_added', 'n_features', 'train_r2', '


plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(results_df['n_features'], results_df['train_r2'], 'b-o', labe
plt.plot(results_df['n_features'], results_df['test_r2'], 'r-o', label
plt.xlabel('Number of Features')
plt.ylabel('R² Score')
plt.title('Forward Stepwise Selection: R² vs Number of Features')
plt.legend()
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.plot(results_df['n_features'], results_df['train_mse'], 'b-o', lab
plt.plot(results_df['n_features'], results_df['test_mse'], 'r-o', labe
plt.xlabel('Number of Features')
plt.ylabel('Mean Squared Error')
plt.title('Forward Stepwise Selection: MSE vs Number of Features')
plt.legend()
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()


best_idx = results_df['test_r2'].idxmax()
best_model_info = results_df.iloc[best_idx]
selected_features_final = best_model_info['features']

print(f"\nBest model uses {len(selected_features_final)} features:")
print("Selected features:", selected_features_final)
print(f"Test R²: {best_model_info['test_r2']:.4f}")
print(f"Test MSE: {best_model_info['test_mse']:.2f}")
```

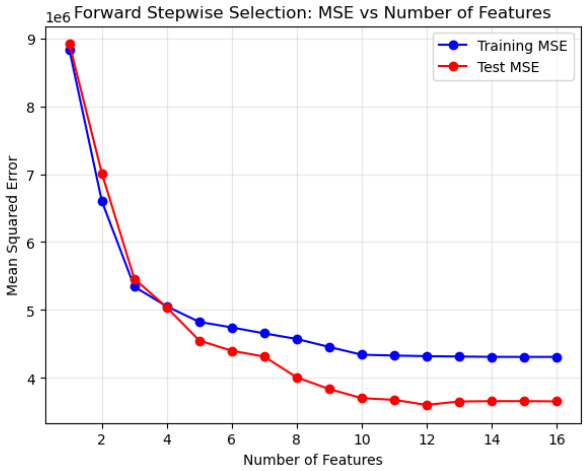
```
Target variable: Outstate
Number of features: 16
Features: ['Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F.Und
ergrad', 'P.Undergrad', 'Room.Board', 'Books', 'Personal', 'PhD', 'Term
inal', 'S.F.Ratio', 'perc.alumni', 'Expend', 'Grad.Rate']

Training set size: 621 samples
Test set size: 156 samples
Starting Forward Stepwise Selection...
==================================================
Step 1: Added 'Expend'
```

```
  Features: ['Expend']
  Train R²: 0.4520, Test R²: 0.4539
  Train MSE: 8831504.70, Test MSE: 8917196.99

Step 2: Added 'Room.Board'
  Features: ['Expend', 'Room.Board']
  Train R²: 0.5905, Test R²: 0.5706
  Train MSE: 6600171.65, Test MSE: 7012611.43

Step 3: Added 'perc.alumni'
  Features: ['Expend', 'Room.Board', 'perc.alumni']
  Train R²: 0.6686, Test R²: 0.6657
  Train MSE: 5341715.42, Test MSE: 5458709.25

Step 4: Added 'Grad.Rate'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate']
  Train R²: 0.6866, Test R²: 0.6916
  Train MSE: 5051241.02, Test MSE: 5035561.39

Step 5: Added 'F.Undergrad'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad']
  Train R²: 0.7008, Test R²: 0.7215
  Train MSE: 4822820.82, Test MSE: 4548065.61

Step 6: Added 'Terminal'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal']
  Train R²: 0.7058, Test R²: 0.7307
  Train MSE: 4740965.53, Test MSE: 4397704.64

Step 7: Added 'S.F.Ratio'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio']
  Train R²: 0.7113, Test R²: 0.7360
  Train MSE: 4653581.80, Test MSE: 4311299.84

Step 8: Added 'Accept'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept']
  Train R²: 0.7164, Test R²: 0.7547
  Train MSE: 4571441.53, Test MSE: 4006102.81

Step 9: Added 'Apps'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps']
  Train R²: 0.7237, Test R²: 0.7652
  Train MSE: 4453664.81, Test MSE: 3834071.67

Step 10: Added 'Top10perc'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc']
```

```
  Train R²: 0.7308, Test R²: 0.7735
  Train MSE: 4339173.58, Test MSE: 3699141.96

Step 11: Added 'Enroll'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l']
  Train R²: 0.7315, Test R²: 0.7750
  Train MSE: 4327076.02, Test MSE: 3674499.88

Step 12: Added 'Personal'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l', 'Personal']
  Train R²: 0.7321, Test R²: 0.7796
  Train MSE: 4317677.54, Test MSE: 3599874.65

Step 13: Added 'P.Undergrad'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l', 'Personal', 'P.Undergrad']
  Train R²: 0.7324, Test R²: 0.7765
  Train MSE: 4312500.38, Test MSE: 3649753.36

Step 14: Added 'Books'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l', 'Personal', 'P.Undergrad', 'Books']
  Train R²: 0.7327, Test R²: 0.7762
  Train MSE: 4307515.04, Test MSE: 3655089.94

Step 15: Added 'PhD'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l', 'Personal', 'P.Undergrad', 'Books', 'PhD']
  Train R²: 0.7328, Test R²: 0.7762
  Train MSE: 4306971.19, Test MSE: 3655418.11

Step 16: Added 'Top25perc'
  Features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate', 'F.Und
ergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc', 'Enrol
l', 'Personal', 'P.Undergrad', 'Books', 'PhD', 'Top25perc']
  Train R²: 0.7328, Test R²: 0.7764
  Train MSE: 4306771.73, Test MSE: 3651563.71

Forward Stepwise Selection Results:
==================================================
    step feature_added  n_features  train_r2   test_r2
0      1        Expend           1  0.452028  0.453940
1      2    Room.Board           2  0.590477  0.570571
2      3   perc.alumni           3  0.668561  0.665727
3      4     Grad.Rate           4  0.686584  0.691639
```

```
4      5   F.Undergrad           5  0.700757  0.721491
5      6      Terminal           6  0.705836  0.730699
6      7     S.F.Ratio           7  0.711257  0.735990
7      8        Accept           8  0.716354  0.754679
8      9          Apps           9  0.723662  0.765214
9     10     Top10perc          10  0.730766  0.773477
10    11        Enroll          11  0.731516  0.774986
11    12      Personal          12  0.732099  0.779555
12    13   P.Undergrad          13  0.732421  0.776501
13    14         Books          14  0.732730  0.776174
14    15           PhD          15  0.732764  0.776154
15    16     Top25perc          16  0.732776  0.776390
```






```
Best model uses 12 features:
Selected features: ['Expend', 'Room.Board', 'perc.alumni', 'Grad.Rate',
'F.Undergrad', 'Terminal', 'S.F.Ratio', 'Accept', 'Apps', 'Top10perc',
'Enroll', 'Personal']
Test R²: 0.7796
Test MSE: 3599874.65
```

10(b)

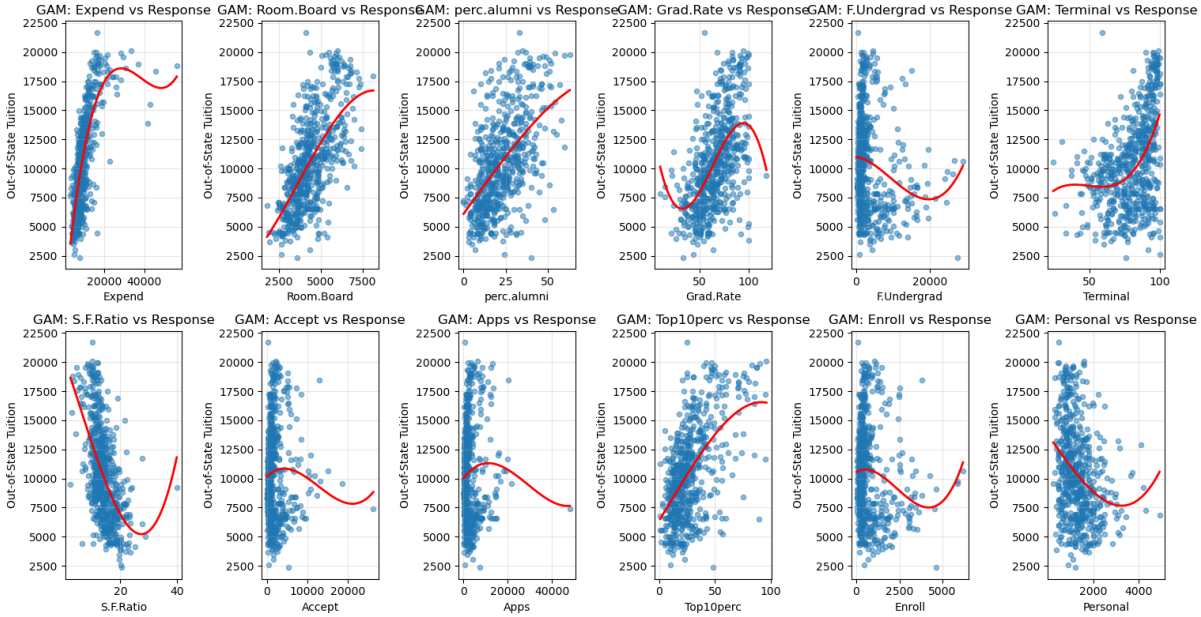
In [57]:

```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Ridge

print("\n" + "=" * 70)
print("PART (B): Fitting GAM with selected features")
print("=" * 70)

def create_gam_features(X, degree=3):
    """
    Create GAM-like features using polynomial basis functions
    """
    gam_features = pd.DataFrame()
    feature_info = {}

    for i, col in enumerate(X.columns):
        # Standardize the feature
```

```
        scaler = StandardScaler()
        x_scaled = scaler.fit_transform(X[[col]]).flatten()

        for deg in range(1, degree + 1):
            feature_name = f"{col}_poly_{deg}"
            gam_features[feature_name] = x_scaled ** deg

        feature_info[col] = {
            'scaler': scaler,
            'start_idx': i * degree,
            'end_idx': (i + 1) * degree
        }

    return gam_features, feature_info


X_train_selected = X_train[selected_features_final]
X_test_selected = X_test[selected_features_final]


X_train_gam, feature_info = create_gam_features(X_train_selected, degr
X_test_gam, _ = create_gam_features(X_test_selected, degree=3)


gam_model = Ridge(alpha=1.0)  # Small regularization
gam_model.fit(X_train_gam, y_train)


y_train_pred_gam = gam_model.predict(X_train_gam)
y_test_pred_gam = gam_model.predict(X_test_gam)


gam_train_r2 = r2_score(y_train, y_train_pred_gam)
gam_test_r2 = r2_score(y_test, y_test_pred_gam)
gam_train_mse = mean_squared_error(y_train, y_train_pred_gam)
gam_test_mse = mean_squared_error(y_test, y_test_pred_gam)

print(f"GAM Model Performance:")
print(f"Training R²: {gam_train_r2:.4f}")
print(f"Test R²: {gam_test_r2:.4f}")
print(f"Training MSE: {gam_train_mse:.2f}")
print(f"Test MSE: {gam_test_mse:.2f}")


n_features = len(selected_features_final)
fig, axes = plt.subplots(2, (n_features + 1) // 2, figsize=(15, 8))
if n_features == 1:
    axes = [axes]
axes = axes.flatten()

for i, feature in enumerate(selected_features_final):
```

```
    x_vals = X_train_selected[feature].values


    x_smooth = np.linspace(x_vals.min(), x_vals.max(), 100)


    simple_model = LinearRegression()


    poly_features = PolynomialFeatures(degree=3, include_bias=False)
    X_poly = poly_features.fit_transform(x_vals.reshape(-1, 1))
    X_smooth_poly = poly_features.transform(x_smooth.reshape(-1, 1))


    simple_model.fit(X_poly, y_train)
    y_smooth = simple_model.predict(X_smooth_poly)


    axes[i].scatter(x_vals, y_train, alpha=0.5, s=20)
    axes[i].plot(x_smooth, y_smooth, 'r-', linewidth=2)
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Out-of-State Tuition')
    axes[i].set_title(f'GAM: {feature} vs Response')
    axes[i].grid(True, alpha=0.3)


for j in range(i + 1, len(axes)):
    axes[j].set_visible(False)

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

```
======================================================================
PART (B): Fitting GAM with selected features
======================================================================
GAM Model Performance:
Training R²: 0.7832
Test R²: 0.7962
Training MSE: 3493365.52
Test MSE: 3327510.89
```



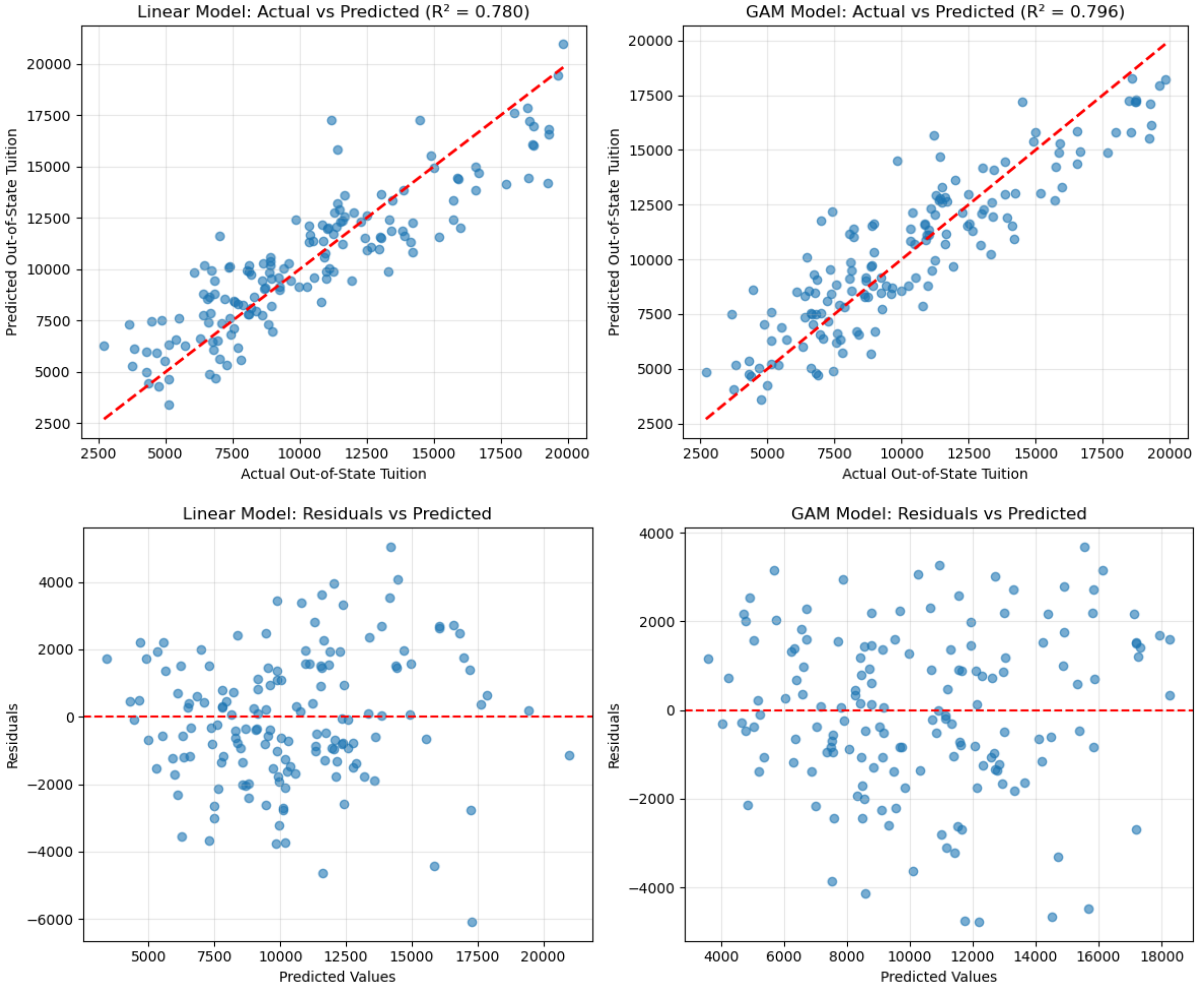
10(c)

In [60]:

```
print("\n" + "=" * 70)
print("PART (C): Model Evaluation on Test Set")
print("=" * 70)


linear_model = LinearRegression()
linear_model.fit(X_train_selected, y_train)
y_test_pred_linear = linear_model.predict(X_test_selected)

linear_test_r2 = r2_score(y_test, y_test_pred_linear)
linear_test_mse = mean_squared_error(y_test, y_test_pred_linear)

print("Test Set Performance Comparison:")
print("-" * 40)
print(f"Linear Model:")
print(f"  R²: {linear_test_r2:.4f}")
print(f"  MSE: {linear_test_mse:.2f}")
print(f"  RMSE: {np.sqrt(linear_test_mse):.2f}")

print(f"\nGAM Model:")
print(f"  R²: {gam_test_r2:.4f}")
print(f"  MSE: {gam_test_mse:.2f}")
print(f"  RMSE: {np.sqrt(gam_test_mse):.2f}")

improvement = ((linear_test_mse - gam_test_mse) / linear_test_mse) * 1
print(f"\nGAM improvement over linear: {improvement:.2f}% reduction in


plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
```

```
plt.scatter(y_test, y_test_pred_linear, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '
plt.xlabel('Actual Out-of-State Tuition')
plt.ylabel('Predicted Out-of-State Tuition')
plt.title(f'Linear Model: Actual vs Predicted (R² = {linear_test_r2:.3
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_test_pred_gam, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '
plt.xlabel('Actual Out-of-State Tuition')
plt.ylabel('Predicted Out-of-State Tuition')
plt.title(f'GAM Model: Actual vs Predicted (R² = {gam_test_r2:.3f})')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()


residuals_linear = y_test - y_test_pred_linear
residuals_gam = y_test - y_test_pred_gam

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_test_pred_linear, residuals_linear, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Linear Model: Residuals vs Predicted')
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.scatter(y_test_pred_gam, residuals_gam, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('GAM Model: Residuals vs Predicted')
plt.grid(True, alpha=0.3)

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

```
======================================================================
PART (C): Model Evaluation on Test Set
======================================================================
Test Set Performance Comparison:
--------------------------------------
Linear Model:
  R²: 0.7796
  MSE: 3599874.65
  RMSE: 1897.33

GAM Model:
  R²: 0.7962
  MSE: 3327510.89
  RMSE: 1824.15

GAM improvement over linear: 7.57% reduction in MSE
```



10(d)

In [65]:
```
print("\n" + "=" * 70)
print("PART (D): Evidence of Non-linear Relationships")
print("=" * 70)
```

```
nonlinear_evidence = {}

for feature in selected_features_final:
    X_feature = X_train_selected[[feature]]
    linear_simple = LinearRegression()
    linear_simple.fit(X_feature, y_train)
    y_pred_linear_simple = linear_simple.predict(X_feature)
    r2_linear = r2_score(y_train, y_pred_linear_simple)

    poly_features = PolynomialFeatures(degree=3, include_bias=False)
    X_poly = poly_features.fit_transform(X_feature)
    poly_simple = LinearRegression()
    poly_simple.fit(X_poly, y_train)
    y_pred_poly_simple = poly_simple.predict(X_poly)
    r2_poly = r2_score(y_train, y_pred_poly_simple)

    improvement = r2_poly - r2_linear
    nonlinear_evidence[feature] = {
        'linear_r2': r2_linear,
        'poly_r2': r2_poly,
        'improvement': improvement,
        'evidence_strength': 'Strong' if improvement > 0.05 else 'Mode
    }

    print(f"{feature}:")
    print(f"  Linear R²: {r2_linear:.4f}")
    print(f"  Polynomial R²: {r2_poly:.4f}")
    print(f"  Improvement: {improvement:.4f}")
    print(f"  Non-linear evidence: {nonlinear_evidence[feature]['evide
    print()

print("Summary of Non-linear Evidence:")
print("-" * 40)
strong_nonlinear = [k for k, v in nonlinear_evidence.items() if v['evi
moderate_nonlinear = [k for k, v in nonlinear_evidence.items() if v['e

if strong_nonlinear:
    print(f"Strong evidence of non-linearity: {strong_nonlinear}")
if moderate_nonlinear:
    print(f"Moderate evidence of non-linearity: {moderate_nonlinear}")
if not strong_nonlinear and not moderate_nonlinear:
    print("Limited evidence of strong non-linear relationships in the

print("\nAnalysis Complete!")
print("=" * 70)
```

```
======================================================================
PART (D): Evidence of Non-linear Relationships
======================================================================
Expend:
  Linear R²: 0.4520
  Polynomial R²: 0.6156
  Improvement: 0.1635
```

```
  Non-linear evidence: Strong

Room.Board:
  Linear R²: 0.4331
  Polynomial R²: 0.4367
  Improvement: 0.0035
  Non-linear evidence: Weak

perc.alumni:
  Linear R²: 0.3048
  Polynomial R²: 0.3059
  Improvement: 0.0011
  Non-linear evidence: Weak

Grad.Rate:
  Linear R²: 0.3071
  Polynomial R²: 0.3355
  Improvement: 0.0284
  Non-linear evidence: Moderate

F.Undergrad:
  Linear R²: 0.0380
  Polynomial R²: 0.0438
  Improvement: 0.0059
  Non-linear evidence: Weak

Terminal:
  Linear R²: 0.1665
  Polynomial R²: 0.2170
  Improvement: 0.0506
  Non-linear evidence: Strong

S.F.Ratio:
  Linear R²: 0.2999
  Polynomial R²: 0.3343
  Improvement: 0.0344
  Non-linear evidence: Moderate

Accept:
  Linear R²: 0.0000
  Polynomial R²: 0.0041
  Improvement: 0.0041
  Non-linear evidence: Weak

Apps:
  Linear R²: 0.0032
  Polynomial R²: 0.0093
  Improvement: 0.0061
  Non-linear evidence: Weak

Top10perc:
  Linear R²: 0.3141
```

```
  Polynomial R²: 0.3227
  Improvement: 0.0086
  Non-linear evidence: Weak

Enroll:
  Linear R²: 0.0176
  Polynomial R²: 0.0248
  Improvement: 0.0072
  Non-linear evidence: Weak

Personal:
  Linear R²: 0.0872
  Polynomial R²: 0.0998
  Improvement: 0.0127
  Non-linear evidence: Weak

Summary of Non-linear Evidence:
----------------------------------------
Strong evidence of non-linearity: ['Expend', 'Terminal']
Moderate evidence of non-linearity: ['Grad.Rate', 'S.F.Ratio']

Analysis Complete!
================================================================================
```

In [ ]: