In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, roc_curve, auc, precision_recall_curve
import joblib

def load_results():
    """Load the saved model results."""
    print("Loading model results...")
    dt_results = pd.read_csv('decision_tree_results.csv')
    lr_results = pd.read_csv('linear_regression_results.csv')
    return dt_results, lr_results

def evaluate_classification_model(results):
    """Evaluate the decision tree classifier performance."""
    print("\nDecision Tree Classifier Evaluation:")

    # Create confusion matrix
    cm = confusion_matrix(results['Actual_Congestion'], results['Predicted_Congestion'])

    # Plot confusion matrix
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix - Traffic Congestion Classification')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.savefig('confusion_matrix.png')
    plt.close()

    # Calculate class-wise accuracy
    class_accuracy = {}
    for i in range(4):  # 4 congestion levels
        class_accuracy[f'Class_{i}'] = cm[i, i] / np.sum(cm[i, :])

    print("\nClass-wise Accuracy:")
    for class_name, acc in class_accuracy.items():
        print(f"{class_name}: {acc:.4f}")

    # Calculate misclassification analysis
```

```
    misclassified = results[results['Actual_Congestion'] != results['Predicted_Congestion']]
    print(f"\nTotal Misclassifications: {len(misclassified)}")
    print("\nMisclassification Distribution:")
    print(misclassified.groupby(['Actual_Congestion', 'Predicted_Congestion']).size())

def evaluate_regression_model(results):
    """Evaluate the linear regression model performance."""
    print("\nLinear Regression Model Evaluation:")

    # Calculate error metrics
    mae = np.mean(np.abs(results['Actual_Traffic'] - results['Predicted_Traffic']))
    mape = np.mean(np.abs((results['Actual_Traffic'] - results['Predicted_Traffic']) / results['Ac

    print(f"Mean Absolute Error: {mae:.2f}")
    print(f"Mean Absolute Percentage Error: {mape:.2f}%")

    # Plot actual vs predicted values
    plt.figure(figsize=(10, 6))
    plt.scatter(results['Actual_Traffic'], results['Predicted_Traffic'], alpha=0.5)
    plt.plot([results['Actual_Traffic'].min(), results['Actual_Traffic'].max()],
             [results['Actual_Traffic'].min(), results['Actual_Traffic'].max()],
             'r--', lw=2)
    plt.xlabel('Actual Traffic Count')
    plt.ylabel('Predicted Traffic Count')
    plt.title('Actual vs Predicted Traffic Volume')
    plt.savefig('regression_scatter.png')
    plt.close()

    # Error distribution
    plt.figure(figsize=(10, 6))
    errors = results['Predicted_Traffic'] - results['Actual_Traffic']
    sns.histplot(errors, bins=50)
    plt.title('Prediction Error Distribution')
    plt.xlabel('Prediction Error')
    plt.ylabel('Count')
    plt.savefig('error_distribution.png')
    plt.close()

    # Calculate error percentiles
    error_percentiles = np.percentile(np.abs(errors), [25, 50, 75, 90, 95, 99])
    print("\nError Percentiles:")
    print(f"25th percentile: {error_percentiles[0]:.2f}")
```

```
    print(f"Median error: {error_percentiles[1]:.2f}")
    print(f"75th percentile: {error_percentiles[2]:.2f}")
    print(f"90th percentile: {error_percentiles[3]:.2f}")
    print(f"95th percentile: {error_percentiles[4]:.2f}")
    print(f"99th percentile: {error_percentiles[5]:.2f}")

def analyze_feature_importance():
    """Analyze and visualize feature importance from both models."""
    # Load the models
    dt_model = joblib.load('decision_tree_model.joblib')
    lr_model = joblib.load('linear_regression_model.joblib')

    # Get feature names (from the training script output)
    feature_names = [
        'hour_sin', 'hour_cos', 'is_morning_peak', 'is_evening_peak',
        'is_weekend', 'day_of_week_num', 'month', 'temperature',
        'humidity', 'wind_speed', 'visibility', 'precipitation',
        'weather_severity', 'rolling_avg_3h', 'traffic_density'
    ]

    # Create feature importance plot
    plt.figure(figsize=(12, 6))

    # Decision Tree importance
    importance_dt = pd.DataFrame({
        'feature': feature_names,
        'importance': dt_model.feature_importances_
    }).sort_values('importance', ascending=True)

    plt.subplot(1, 2, 1)
    plt.barh(importance_dt['feature'], importance_dt['importance'])
    plt.title('Decision Tree Feature Importance')

    # Linear Regression coefficients
    importance_lr = pd.DataFrame({
        'feature': feature_names,
        'coefficient': np.abs(lr_model.coef_)
    }).sort_values('coefficient', ascending=True)

    plt.subplot(1, 2, 2)
    plt.barh(importance_lr['feature'], importance_lr['coefficient'])
    plt.title('Linear Regression Feature Importance')
```

```
    plt.tight_layout()
    plt.savefig('feature_importance.png')
    plt.close()

def main():
    # Load results
    dt_results, lr_results = load_results()

    # Evaluate models
    evaluate_classification_model(dt_results)
    evaluate_regression_model(lr_results)

    # Analyze feature importance
    analyze_feature_importance()

    print("\nEvaluation complete. Check the generated visualization files for detailed analysis.")

if __name__ == "__main__":
    main()
```

```
Loading model results...

Decision Tree Classifier Evaluation:

Class-wise Accuracy:
Class_0: 0.8969
Class_1: 0.9794
Class_2: 0.9138
Class_3: 0.9192

Total Misclassifications: 236

Misclassification Distribution:
Actual_Congestion  Predicted_Congestion
0                  2                      12
                   3                      71
1                  2                      16
2                  0                      55
                   1                      17
                   3                       1
3                  0                      64
dtype: int64

Linear Regression Model Evaluation:
Mean Absolute Error: 65.01
Mean Absolute Percentage Error: 14.03%

Error Percentiles:
25th percentile: 22.97
Median error: 49.25
75th percentile: 90.32
90th percentile: 140.49
95th percentile: 183.88
99th percentile: 268.40

Evaluation complete. Check the generated visualization files for detailed analysis.
```

In [ ]: