In [2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, classification_report
import joblib

def load_and_prepare_data(file_path):
    """Load and prepare the engineered dataset."""
    print("Loading engineered data...")
    df = pd.read_csv(file_path)

    # Convert categorical variables
    le = LabelEncoder()
    categorical_columns = ['weather_main', 'visibility_category', 'temp_category', 'congestion_leve

    for col in categorical_columns:
        df[f'{col}_encoded'] = le.fit_transform(df[col])

    return df

def prepare_features(df):
    """Prepare feature sets for both models."""
    print("Preparing features...")

    # Features for both models
    basic_features = [
        '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'
    ]

    # Convert boolean columns to int
    df['is_morning_peak'] = df['is_morning_peak'].astype(int)
    df['is_evening_peak'] = df['is_evening_peak'].astype(int)
```

```
    df['is_weekend'] = df['is_weekend'].astype(int)

    # Separate features for classification and regression
    X_class = df[basic_features]
    X_reg = df[basic_features]
    y_class = df['congestion_level_encoded']
    y_reg = df['traffic_count']

    # Handle missing values
    imputer = SimpleImputer(strategy='mean')
    X_class_imputed = pd.DataFrame(imputer.fit_transform(X_class), columns=X_class.columns)
    X_reg_imputed = pd.DataFrame(imputer.fit_transform(X_reg), columns=X_reg.columns)

    return X_class_imputed, X_reg_imputed, y_class, y_reg

def train_decision_tree(X, y):
    """Train and tune decision tree classifier for congestion levels."""
    print("\nTraining Decision Tree Classifier...")

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Scale features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Define parameter grid
    param_grid = {
        'max_depth': [5, 10, 15, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Create and tune model
    dt_classifier = DecisionTreeClassifier(random_state=42)
    grid_search = GridSearchCV(dt_classifier, param_grid, cv=5, scoring='accuracy')
    grid_search.fit(X_train_scaled, y_train)

    # Get best model
    best_model = grid_search.best_estimator_
```

```
    # Make predictions
    y_pred = best_model.predict(X_test_scaled)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    class_report = classification_report(y_test, y_pred)

    print(f"\nBest parameters: {grid_search.best_params_}")
    print(f"Accuracy: {accuracy:.4f}")
    print("\nClassification Report:")
    print(class_report)

    # Feature importance
    feature_importance = pd.DataFrame({
        'feature': X.columns,
        'importance': best_model.feature_importances_
    }).sort_values('importance', ascending=False)

    print("\nTop 10 Most Important Features for Classification:")
    print(feature_importance.head(10))

    # Save model and scaler
    joblib.dump(best_model, 'decision_tree_model.joblib')
    joblib.dump(scaler, 'dt_scaler.joblib')

    return best_model, scaler, (X_test_scaled, y_test, y_pred)

def train_linear_regression(X, y):
    """Train linear regression model for continuous traffic prediction."""
    print("\nTraining Linear Regression Model...")

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Scale features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Train model
    lr_model = LinearRegression()
    lr_model.fit(X_train_scaled, y_train)
```

```
    # Make predictions
    y_pred = lr_model.predict(X_test_scaled)

    # Calculate metrics
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"\nRegression Metrics:")
    print(f"Mean Squared Error: {mse:.2f}")
    print(f"Root Mean Squared Error: {rmse:.2f}")
    print(f"R-squared Score: {r2:.4f}")

    # Feature importance
    feature_importance = pd.DataFrame({
        'feature': X.columns,
        'coefficient': lr_model.coef_
    }).sort_values('coefficient', key=abs, ascending=False)

    print("\nTop 10 Most Important Features for Regression:")
    print(feature_importance.head(10))

    # Save model and scaler
    joblib.dump(lr_model, 'linear_regression_model.joblib')
    joblib.dump(scaler, 'lr_scaler.joblib')

    return lr_model, scaler, (X_test_scaled, y_test, y_pred)

def save_results(dt_results, lr_results):
    """Save model results and predictions."""
    # Save Decision Tree results
    X_test_dt, y_test_dt, y_pred_dt = dt_results
    dt_results_df = pd.DataFrame({
        'Actual_Congestion': y_test_dt,
        'Predicted_Congestion': y_pred_dt
    })
    dt_results_df.to_csv('decision_tree_results.csv', index=False)

    # Save Linear Regression results
    X_test_lr, y_test_lr, y_pred_lr = lr_results
    lr_results_df = pd.DataFrame({
```

```
        'Actual_Traffic': y_test_lr,
        'Predicted_Traffic': y_pred_lr,
        'Absolute_Error': np.abs(y_test_lr - y_pred_lr)
    })
    lr_results_df.to_csv('linear_regression_results.csv', index=False)

    # Calculate and save error statistics
    error_stats = {
        'Mean_Absolute_Error': np.mean(np.abs(y_test_lr - y_pred_lr)),
        'Median_Absolute_Error': np.median(np.abs(y_test_lr - y_pred_lr)),
        'Max_Absolute_Error': np.max(np.abs(y_test_lr - y_pred_lr))
    }
    pd.DataFrame([error_stats]).to_csv('error_statistics.csv', index=False)

def main():
    # Load and prepare data
    df = load_and_prepare_data('engineered_traffic_data.csv')

    # Prepare features with imputation
    X_class, X_reg, y_class, y_reg = prepare_features(df)

    # Train models
    dt_model, dt_scaler, dt_results = train_decision_tree(X_class, y_class)
    lr_model, lr_scaler, lr_results = train_linear_regression(X_reg, y_reg)

    # Save results
    save_results(dt_results, lr_results)

    print("\nAll models and results have been saved.")

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

```
Loading engineered data...
Preparing features...

Training Decision Tree Classifier...

Best parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Accuracy: 0.9268

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       805
           1       0.98      0.98      0.98       778
           2       0.97      0.91      0.94       847
           3       0.91      0.92      0.91       792

    accuracy                           0.93      3222
   macro avg       0.93      0.93      0.93      3222
weighted avg       0.93      0.93      0.93      3222


Top 10 Most Important Features for Classification:
             feature  importance
14  traffic_density    0.668878
13   rolling_avg_3h    0.198964
0          hour_sin    0.050038
6             month    0.032758
7       temperature    0.016522
1          hour_cos    0.014825
5   day_of_week_num    0.009132
8          humidity    0.003250
9        wind_speed    0.001930
2   is_morning_peak    0.001108

Training Linear Regression Model...

Regression Metrics:
Mean Squared Error: 7623.34
Root Mean Squared Error: 87.31
R-squared Score: 0.9759
```

```
Top 10 Most Important Features for Regression:
            feature  coefficient
13   rolling_avg_3h   328.279361
14  traffic_density   273.338198
0          hour_sin    81.234189
7       temperature    33.552281
4        is_weekend   -26.728984
2   is_morning_peak    20.411655
5   day_of_week_num    16.221122
6             month   -12.815130
10       visibility     8.228632
9        wind_speed    -6.018819

All models and results have been saved.
```

In [ ]: