In [2]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
from scipy.stats.mstats import winsorize
import plotly.express as py
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder, MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import warnings
```

In [4]:
```
file_path = 'cars.csv'
cars_data = pd.read_csv(file_path)
cars_data.info(), cars_data.head()
```

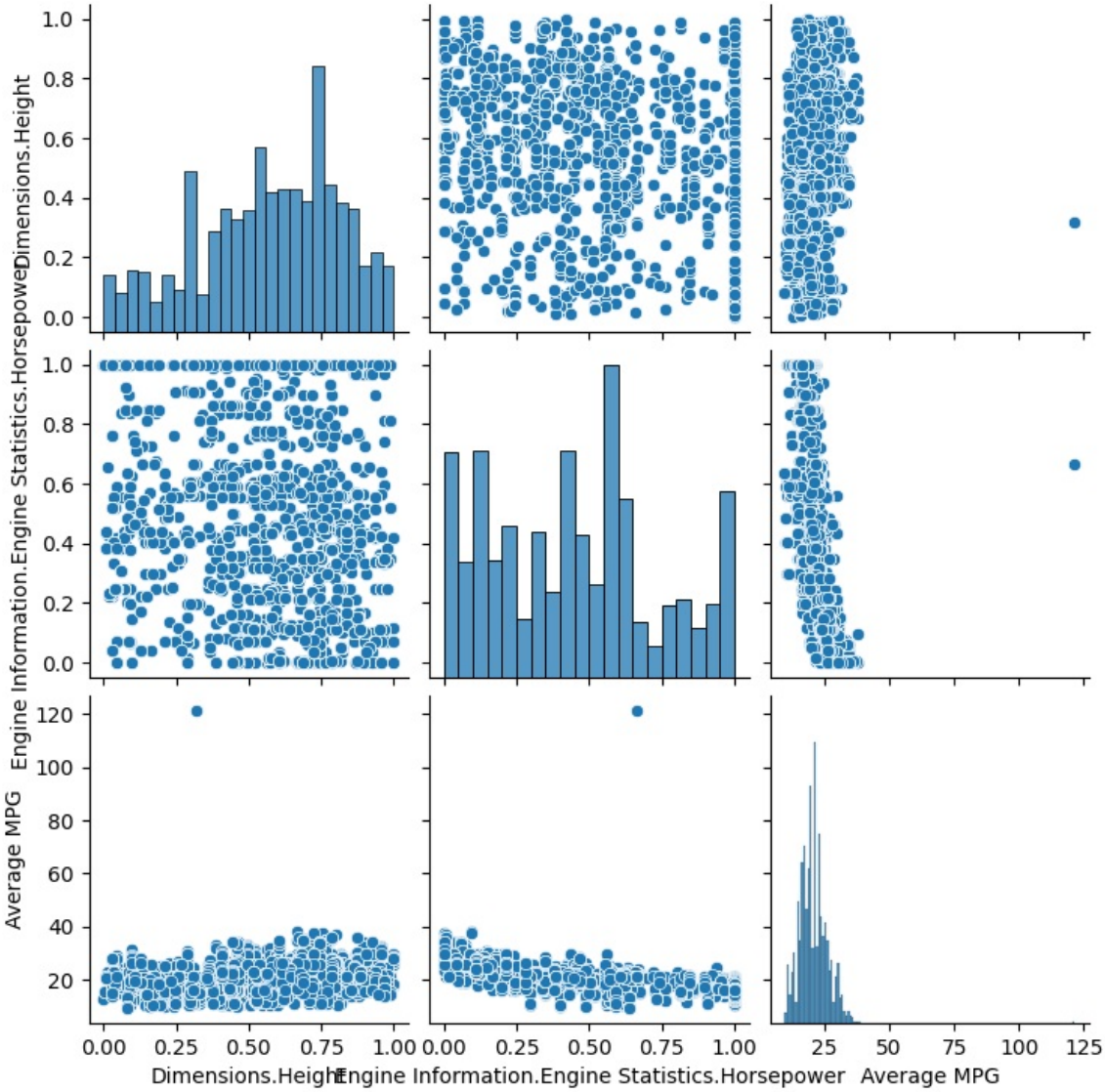
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5076 entries, 0 to 5075
Data columns (total 18 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Dimensions.Height                             5076 non-null   int64
 1   Dimensions.Length                             5076 non-null   int64
 2   Dimensions.Width                              5076 non-null   int64
 3   Engine Information.Driveline                  5076 non-null   object
 4   Engine Information.Engine Type                5076 non-null   object
 5   Engine Information.Hybrid                     5076 non-null   bool
 6   Engine Information.Number of Forward Gears    5076 non-null   int64
 7   Engine Information.Transmission               5076 non-null   object
 8   Fuel Information.City mpg                     5076 non-null   int64
 9   Fuel Information.Fuel Type                    5076 non-null   object
 10  Fuel Information.Highway mpg                  5076 non-null   int64
 11  Identification.Classification                 5076 non-null   object
 12  Identification.ID                             5076 non-null   object
 13  Identification.Make                           5076 non-null   object
 14  Identification.Model Year                     5076 non-null   object
 15  Identification.Year                           5076 non-null   int64
 16  Engine Information.Engine Statistics.Horsepower  5076 non-null   int64
 17  Engine Information.Engine Statistics.Torque   5076 non-null   int64
dtypes: bool(1), int64(9), object(8)
memory usage: 679.2+ KB
```

Out[4]:
```
(None,
   Dimensions.Height  Dimensions.Length  Dimensions.Width  \
0                140                143               202
1                140                143               202
2                140                143               202
3                140                143               202
4                140                143               202

  Engine Information.Driveline               Engine Information.Engine Type  \
0             All-wheel drive          Audi 3.2L 6 cylinder 250hp 236ft-lbs
1           Front-wheel drive  Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo
2           Front-wheel drive  Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo
3             All-wheel drive  Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo
4             All-wheel drive  Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo

   Engine Information.Hybrid  Engine Information.Number of Forward Gears  \
0                       True                                          6
1                       True                                          6
2                       True                                          6
3                       True                                          6
4                       True                                          6

  Engine Information.Transmission  Fuel Information.City mpg  \
0  6 Speed Automatic Select Shift                         18
1  6 Speed Automatic Select Shift                         22
2                6 Speed Manual                           21
3  6 Speed Automatic Select Shift                         21
4  6 Speed Automatic Select Shift                         21

  Fuel Information.Fuel Type  Fuel Information.Highway mpg  \
0                   Gasoline                            25
1                   Gasoline                            28
2                   Gasoline                            30
3                   Gasoline                            28
4                   Gasoline                            28

  Identification.Classification           Identification.ID  \

0        Automatic transmission            2009 Audi A3 3.2
1        Automatic transmission       2009 Audi A3 2.0 T AT
2           Manual transmission          2009 Audi A3 2.0 T
3        Automatic transmission  2009 Audi A3 2.0 T Quattro
4        Automatic transmission  2009 Audi A3 2.0 T Quattro

  Identification.Make Identification.Model Year  Identification.Year  \
0               Audi              2009 Audi A3                2009
1               Audi              2009 Audi A3                2009
2               Audi              2009 Audi A3                2009
3               Audi              2009 Audi A3                2009
4               Audi              2009 Audi A3                2009

   Engine Information.Engine Statistics.Horsepower  \
0                                              250
1                                              200
2                                              200
3                                              200
4                                              200

   Engine Information.Engine Statistics.Torque
0                                          236
1                                          207
2                                          207
3                                          207
4                                          207  )
```

In [5]:
```
warnings.simplefilter(action='ignore', category=FutureWarning)

# Handling Missing Values
cars_data.dropna(inplace=True)

# Dealing with Outliers (Winsorization)
cars_data['Engine Information.Engine Statistics.Horsepower'] = winsorize(cars_data['Engine Information.Engine S

# Standardizing Data Formats
cars_data['Dimensions.Height'] = pd.to_numeric(cars_data['Dimensions.Height'], errors='coerce')

# Converting string columns to integer or create labels
cars_data['Engine Information.Driveline'] = cars_data['Engine Information.Driveline'].map({'All-wheel drive': '
cars_data['Engine Information.Hybrid'] = cars_data['Engine Information.Hybrid'].map({'Yes': 1, 'No': 0})


# Converting string columns to integer or create labels
cars_data['Engine Information.Driveline'] = cars_data['Engine Information.Driveline'].map({'All-wheel drive': '
```

```
cars_data['Engine Information.Hybrid'] = cars_data['Engine Information.Hybrid'].map({'Yes': 1, 'No': 0})


# Feature Engineering (Creating a new feature)
cars_data['Average MPG'] = (cars_data['Fuel Information.City mpg'] + cars_data['Fuel Information.Highway mpg'])
cars_data['Engine Information.Torque per Horsepower'] = cars_data['Engine Information.Engine Statistics.Torque'

# Data Transformation (Scaling numeric features)
scaler = MinMaxScaler()
cars_data[['Dimensions.Height', 'Engine Information.Engine Statistics.Horsepower']] = scaler.fit_transform(cars_

# Replace infinite values with NaN
cars_data.replace([np.inf, -np.inf], np.nan, inplace=True)

# Data Validation (Checking for duplicates)
print("Number of duplicate rows:", cars_data.duplicated().sum())
cars_data.drop_duplicates(inplace=True)

# Data Visualization (Pairplot)
sns.pairplot(cars_data[['Dimensions.Height', 'Engine Information.Engine Statistics.Horsepower', 'Average MPG']]
with warnings.catch_warnings():
    warnings.simplefilter("ignore", category=FutureWarning)
```

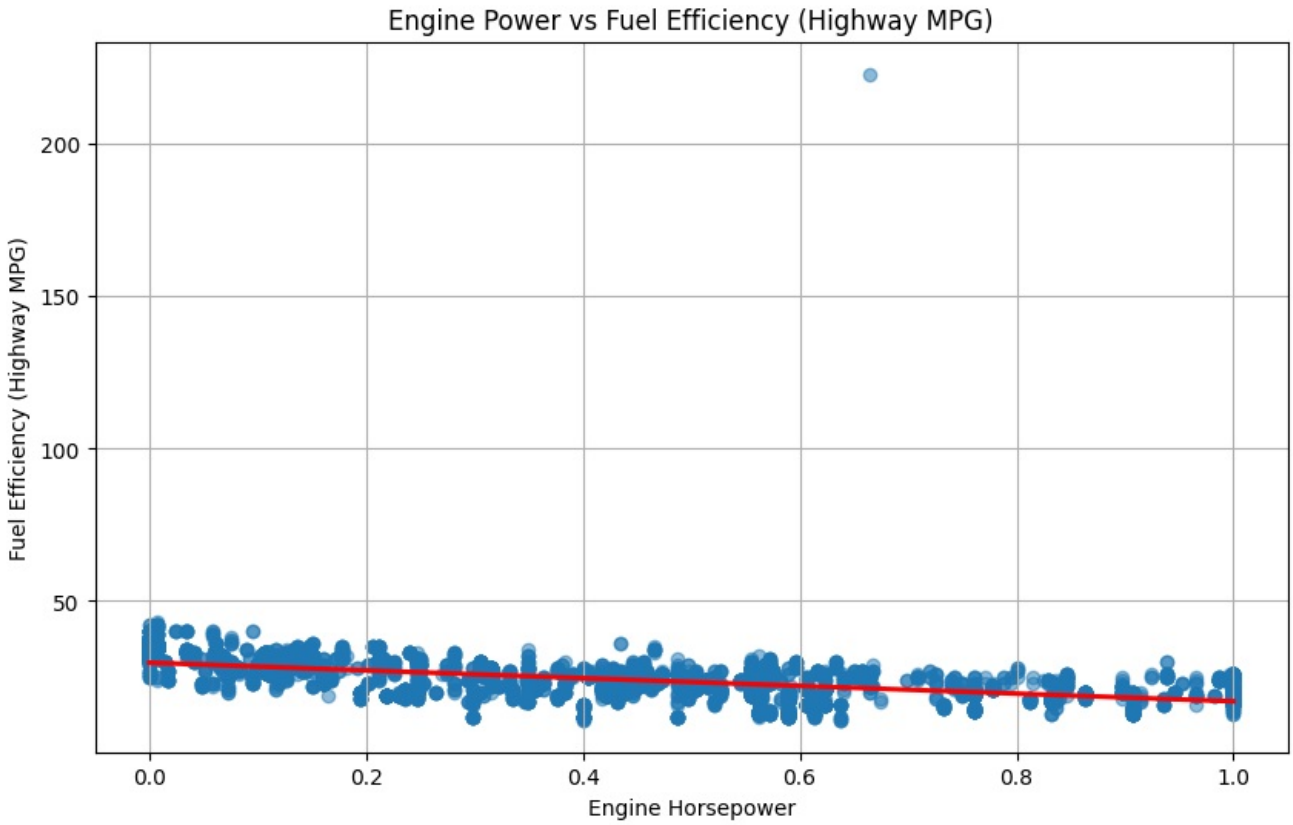
```
Number of duplicate rows: 18
```



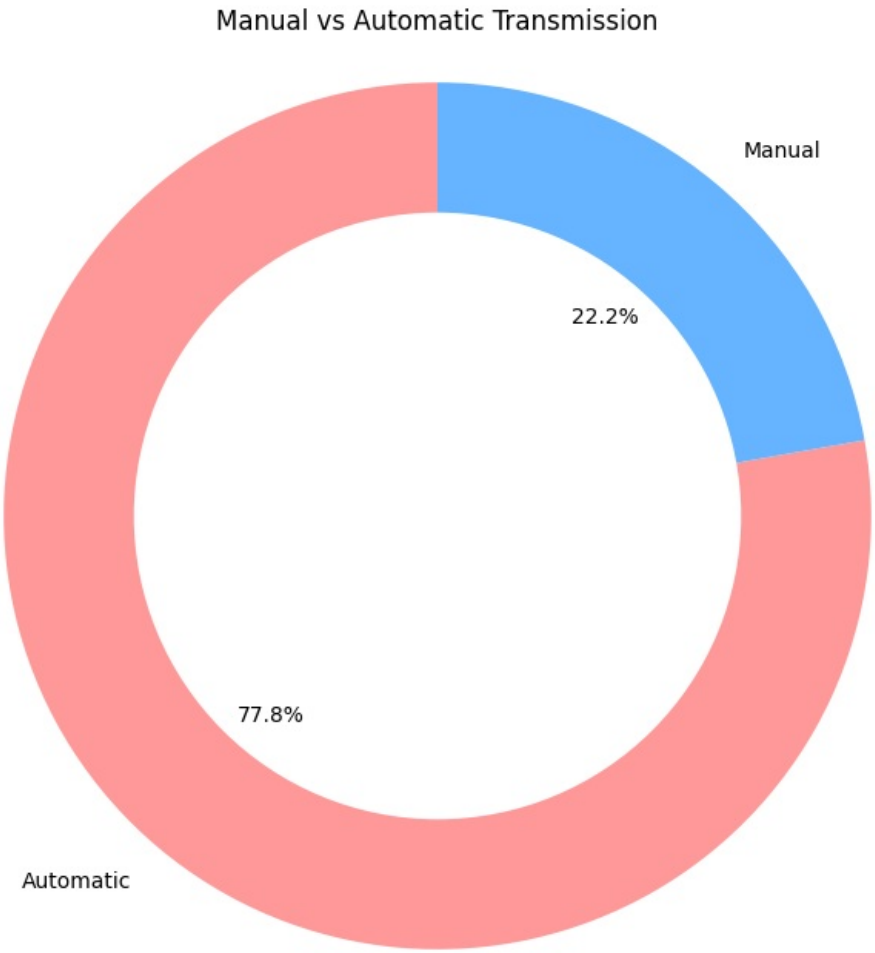
In [9]:
```
x = cars_data['Engine Information.Engine Statistics.Horsepower']
y = cars_data['Fuel Information.Highway mpg']

regression = linregress(x, y)
r_squared = regression.rvalue ** 2
```

In [10]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x=x, y=y, scatter_kws={'alpha':0.5}, line_kws={"color": "red"})
plt.title('Engine Power vs Fuel Efficiency (Highway MPG)')
plt.xlabel('Engine Horsepower')
plt.ylabel('Fuel Efficiency (Highway MPG)')
```

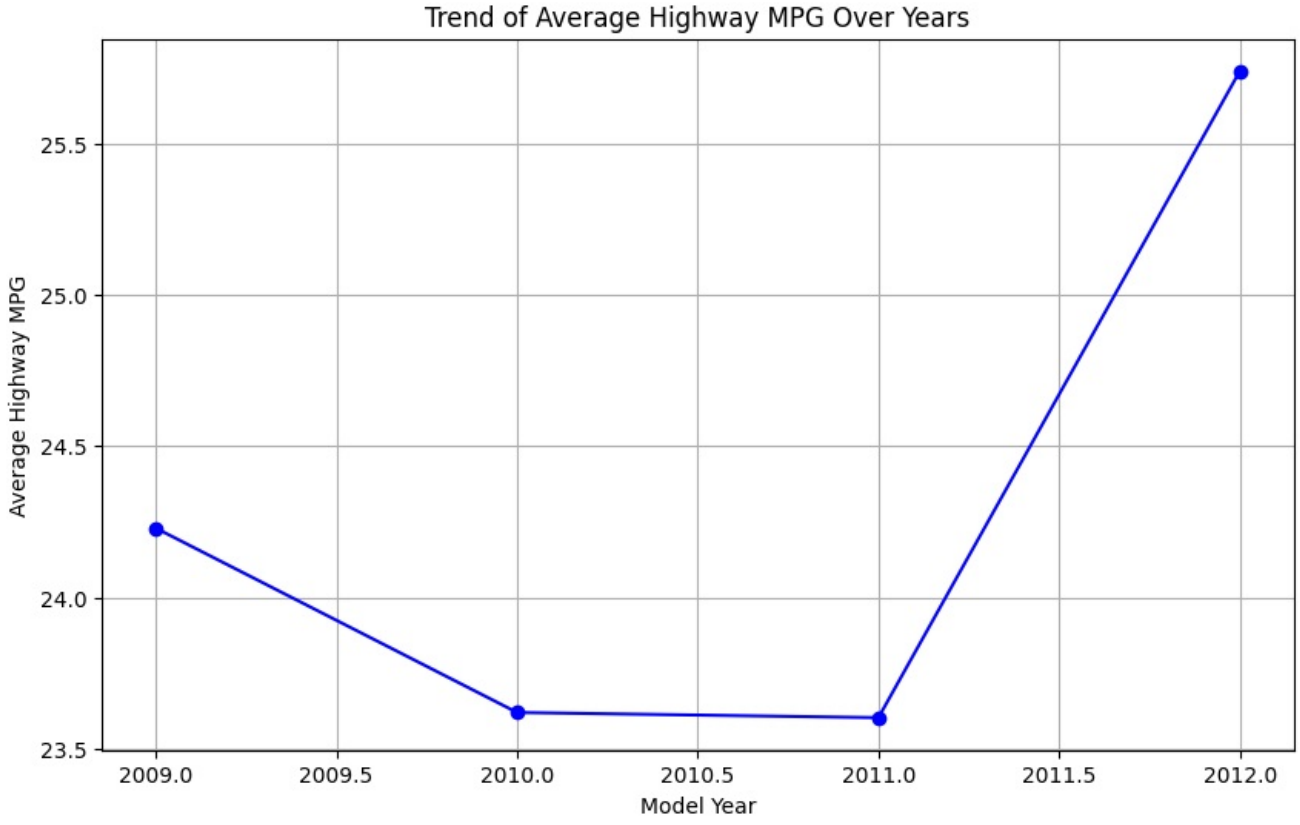
```
plt.grid(True)
```



In [11]:
```
transmission_counts = cars_data['Engine Information.Transmission'].apply(lambda x: 'Automatic' if 'Automatic' i

plt.figure(figsize=(8, 8))
plt.pie(transmission_counts, labels=transmission_counts.index, autopct='%1.1f%%', startangle=90, colors=['#ff99
plt.title('Manual vs Automatic Transmission')
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.axis('equal')
plt.show()
```



In [55]:
```
yearly_highway_mpg = cars_data.groupby('Identification.Year')['Fuel Information.Highway mpg'].mean().reset_index
plt.figure(figsize=(10, 6))
plt.plot(yearly_highway_mpg['Identification.Year'], yearly_highway_mpg['Fuel Information.Highway mpg'], marker=
plt.title('Trend of Average Highway MPG Over Years')
plt.xlabel('Model Year')
plt.ylabel('Average Highway MPG')
plt.grid(True)
plt.show()
```



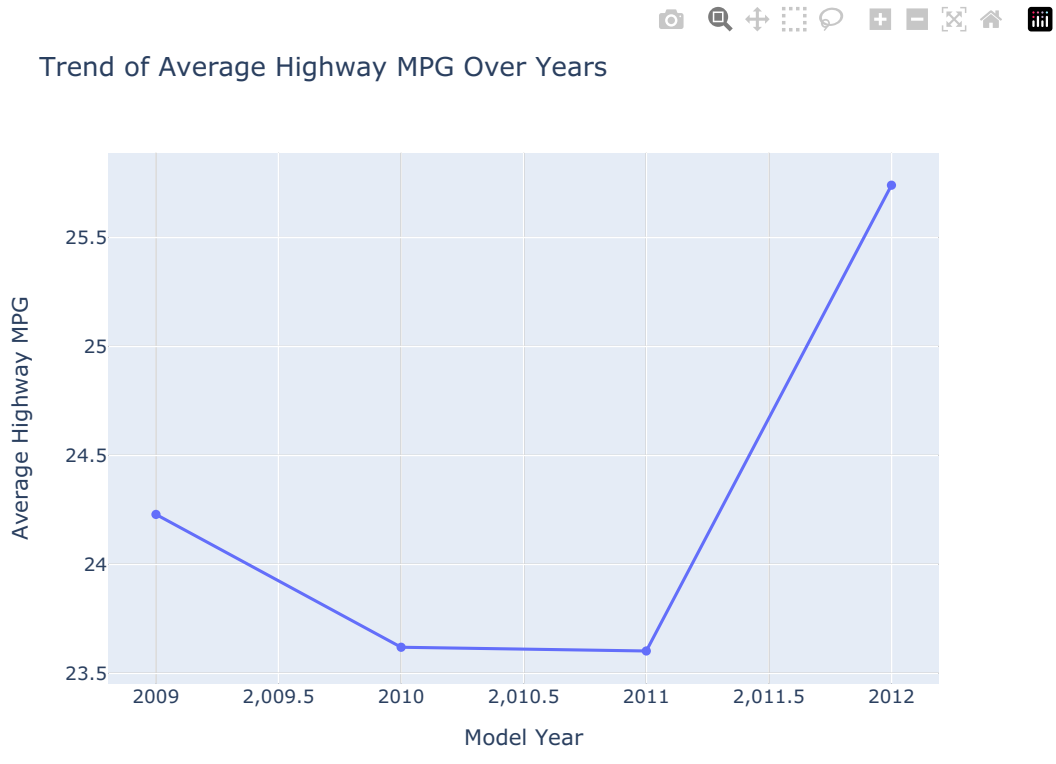
In [56]:
```
categorical_features = ['Engine Information.Driveline', 'Fuel Information.Fuel Type']
numerical_features = ['Engine Information.Engine Statistics.Horsepower',
```

```
                      'Engine Information.Engine Statistics.Torque',
                      'Engine Information.Number of Forward Gears',
                      'Dimensions.Width',
                      'Dimensions.Height']
target = cars_data['Fuel Information.City mpg']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
model_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])
X_train, X_test, y_train, y_test = train_test_split(cars_data[numerical_features + categorical_features], targe
model_pipeline.fit(X_train, y_train)
predictions = model_pipeline.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print("Mean Absolute Error:", mae)
print("R^2 Score:", r2)
```

```
Mean Absolute Error: 0.42972198977654946
R^2 Score: 0.9721816853473549
```

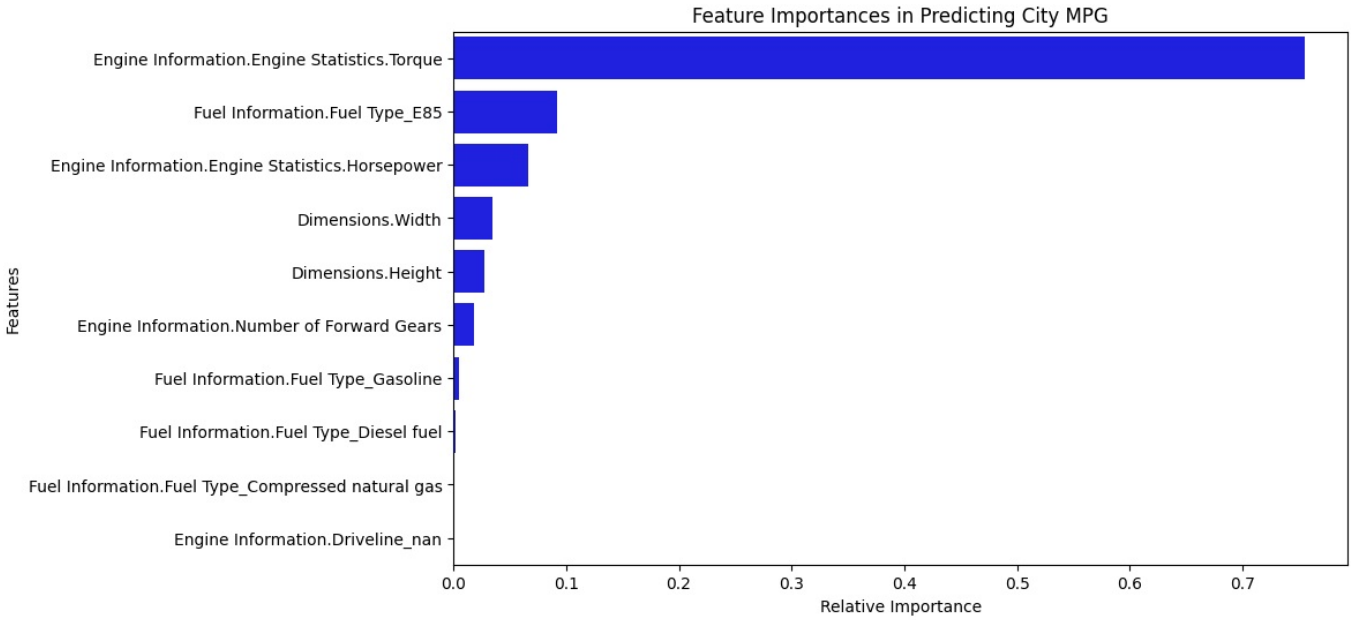
In [14]:

```
yearly_highway_mpg = cars_data.groupby('Identification.Year').agg({
    'Fuel Information.Highway mpg': 'mean'
}).reset_index()

yearly_highway_mpg.columns = ['Model Year', 'Average Highway MPG']

fig = py.line(yearly_highway_mpg, x='Model Year', y='Average Highway MPG',
              title='Trend of Average Highway MPG Over Years',
              labels={'Model Year': 'Model Year', 'Average Highway MPG': 'Average Highway MPG'},
              markers=True)
fig.show()
```

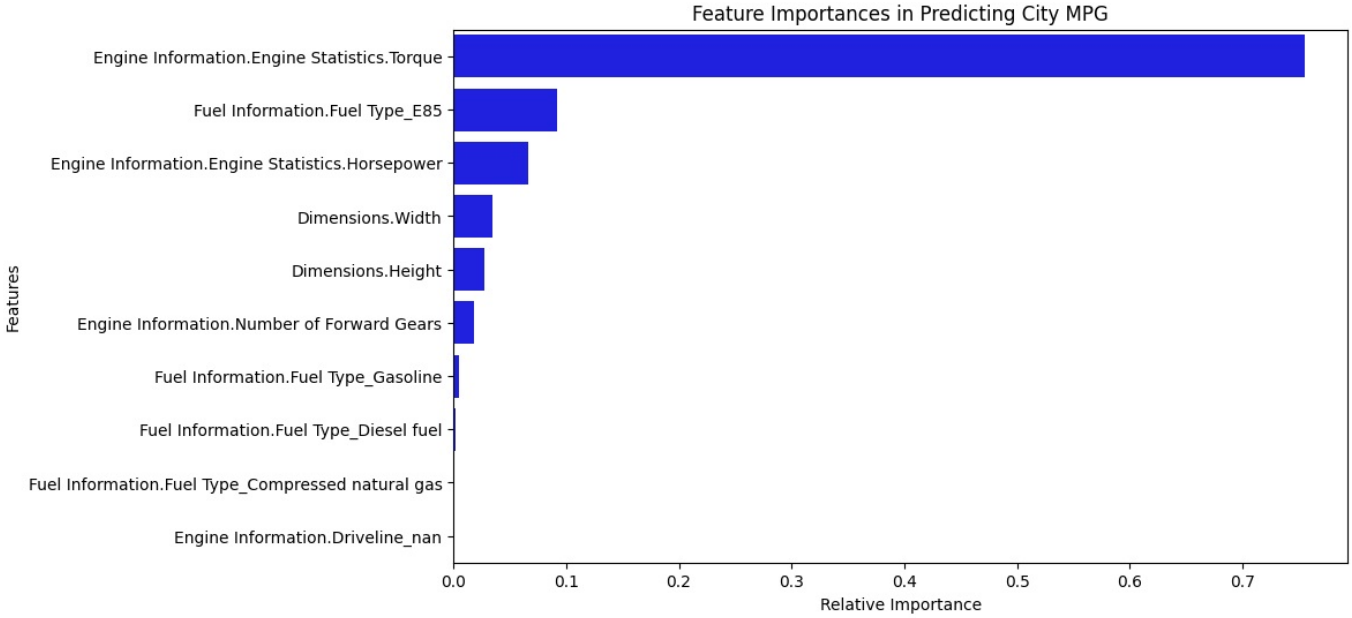


In [58]:

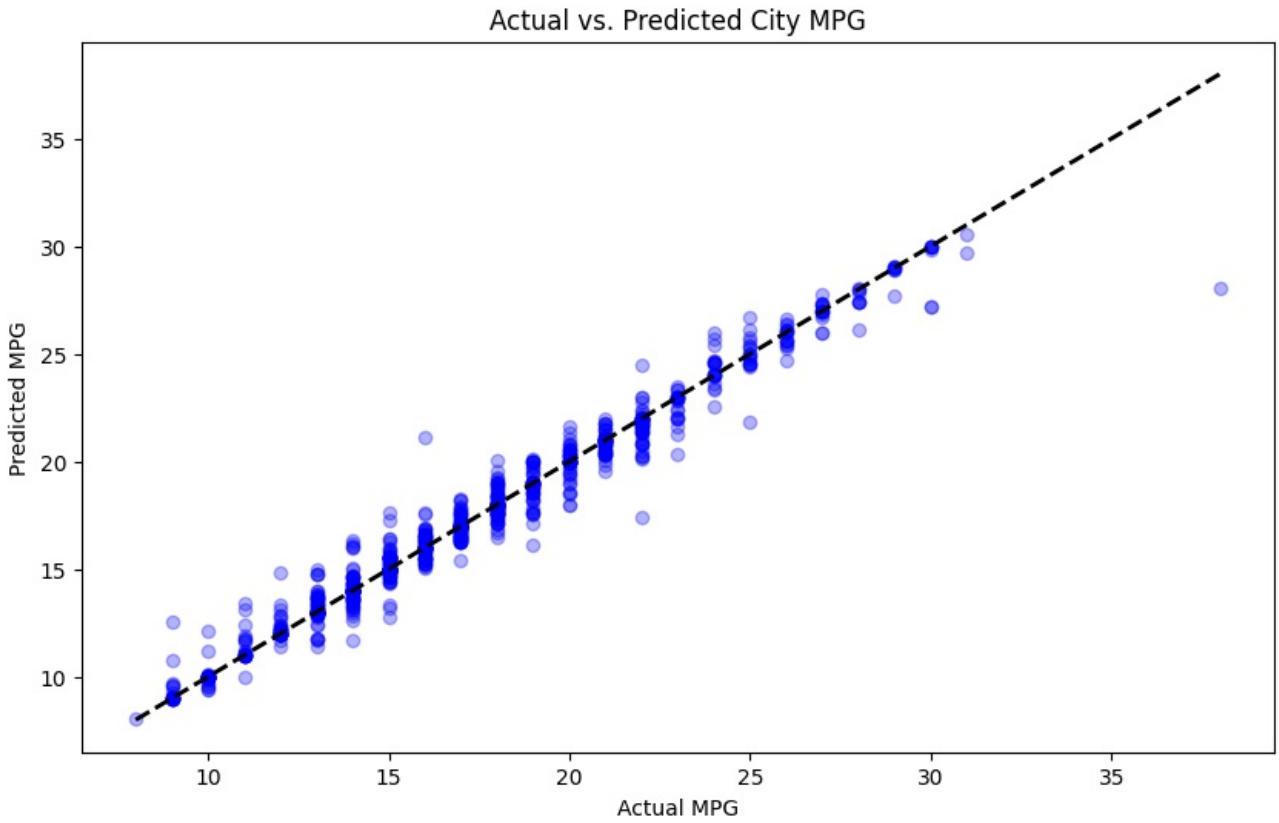
```
# Feature Importance
feature_names = (numerical_features +
                 list(model_pipeline.named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(cate
importances = model_pipeline.named_steps['regressor'].feature_importances_
indices = np.argsort(importances)[::-1]

# Plotting Feature Importances
plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=np.array(feature_names)[indices], color='b')
plt.title('Feature Importances in Predicting City MPG')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
plt.show()
```



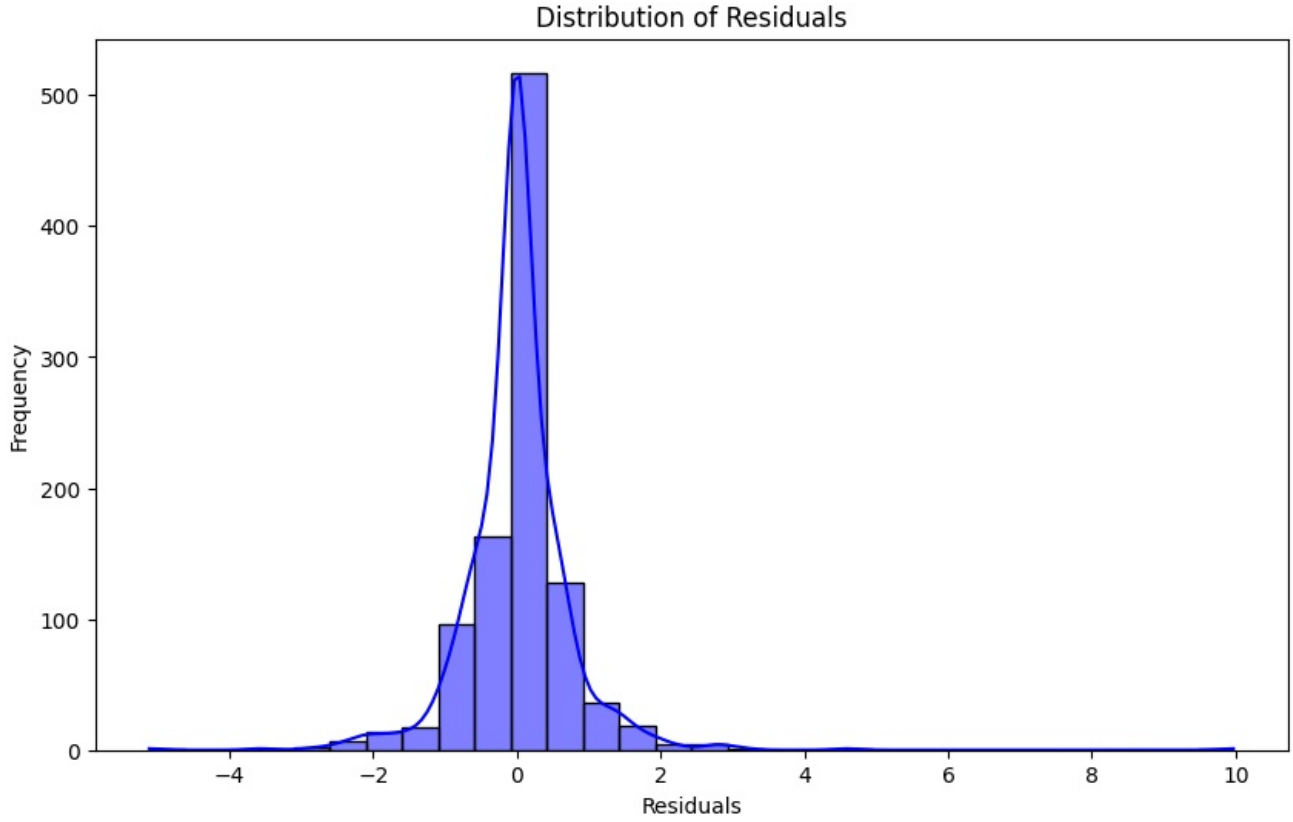
In [59]:
```
feature_names = (numerical_features +
                 list(model_pipeline.named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(cate
importances = model_pipeline.named_steps['regressor'].feature_importances_
indices = np.argsort(importances)[::-1]
plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=np.array(feature_names)[indices], color='b')
plt.title('Feature Importances in Predicting City MPG')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
plt.show()
```



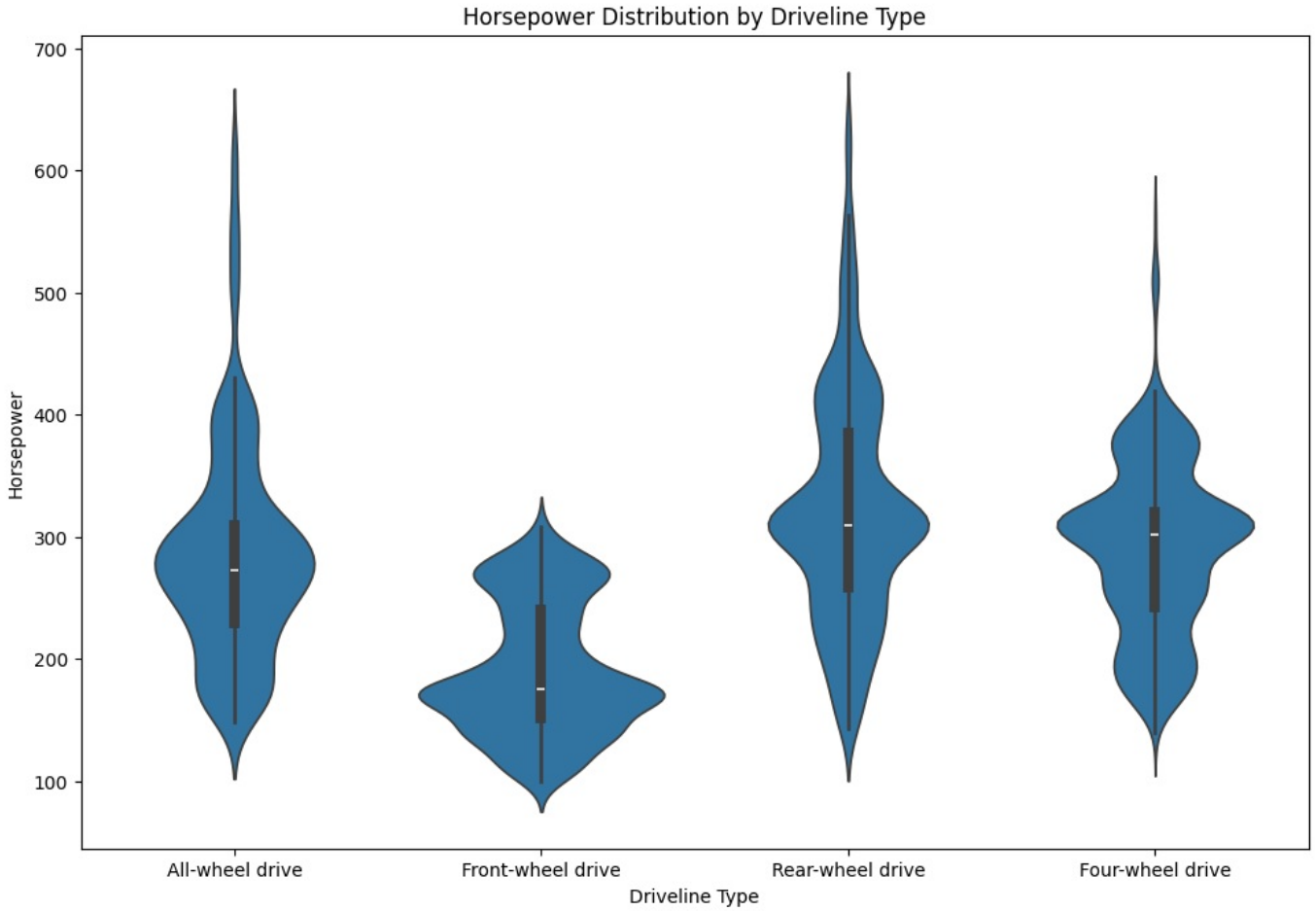
In [60]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predictions, alpha=0.3, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)  # Diagonal line
plt.title('Actual vs. Predicted City MPG')
plt.xlabel('Actual MPG')
plt.ylabel('Predicted MPG')
plt.show()
```



In [61]:
```
residuals = y_test - predictions
plt.figure(figsize=(10, 6))
sns.histplot(residuals, bins=30, kde=True, color='blue')
plt.title('Distribution of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
```



In [7]:
```
cars_data['Horsepower'] = cars_data['Engine Information.Engine Statistics.Horsepower']
cars_data['Driveline'] = cars_data['Engine Information.Driveline']

plt.figure(figsize=(12, 8))
sns.violinplot(x='Driveline', y='Horsepower', data=cars_data)
plt.title('Horsepower Distribution by Driveline Type')
plt.xlabel('Driveline Type')
plt.ylabel('Horsepower')
plt.show()
```



In [41]:
```
plt.figure(figsize=(10, 6))
plt.hist(cars_data['Engine Information.Number of Forward Gears'], bins=range(1, cars_data['Engine Information.Nu
plt.title('Distribution of Number of Forward Gears in Vehicles')
plt.xlabel('Number of Forward Gears')
plt.ylabel('Number of Vehicles')
plt.xticks(range(1, cars_data['Engine Information.Number of Forward Gears'].max() + 1))  # Ensure ticks for eve
plt.grid(axis='y', linestyle='--')
plt.show()
```