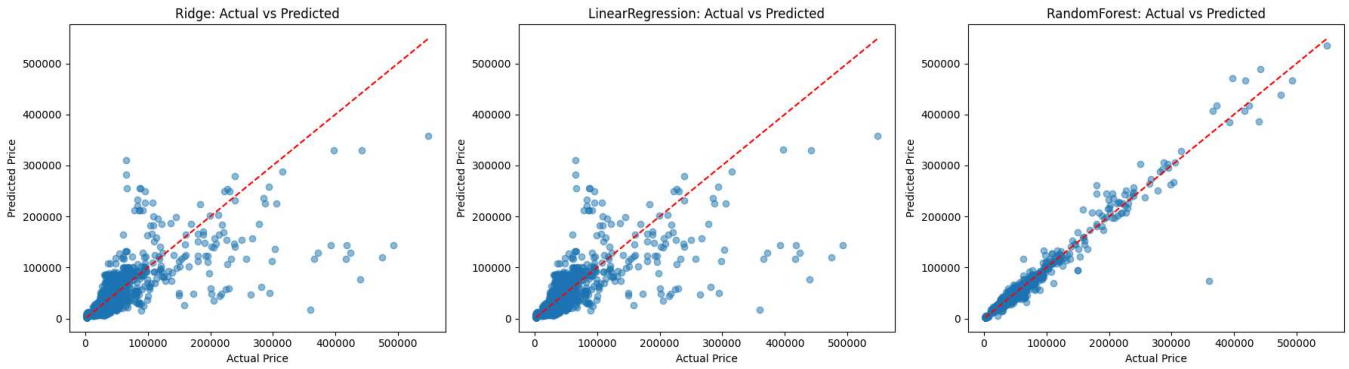
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error


# Load dataset
car_data = pd.read_csv('/content/data.csv')


# Clean column names
car_data.columns = car_data.columns.str.lower().str.replace(' ', '_')
string_columns = list(car_data.dtypes[car_data.dtypes == 'object'].index)
for col in string_columns:
    car_data[col] = car_data[col].str.lower().str.replace(' ', '_')
car_data.rename(columns={'msrp': 'price'}, inplace=True)


# Log transformation of prices
car_data['log_price'] = np.log1p(car_data['price'])


# Train-test split
np.random.seed(2)
n = len(car_data)
n_val = int(n * 0.2)
n_test = int(n * 0.2)
n_train = n - n_val - n_test
idx = np.arange(n)
np.random.shuffle(idx)
car_data_shuffled = car_data.iloc[idx]
car_data_train = car_data_shuffled.iloc[:n_train].copy()
car_data_val = car_data_shuffled.iloc[n_train:n_train+n_val].copy()
car_data_test = car_data_shuffled.iloc[n_train+n_val:].copy()

y_train = car_data_train.log_price.values
y_val = car_data_val.log_price.values


# Features
base = ['engine_hp', 'engine_cylinders', 'highway_mpg', 'city_mpg', 'popularity']

def prepare_X(df):
    df = df.copy()
    df['age'] = 2023 - df['year']
    features = base + ['age']
    df = df[features].fillna(df[features].mean())
    return df.values

x_train = prepare_X(car_data_train)
x_val = prepare_X(car_data_val)



# Models
models = {
    "Ridge": Ridge(alpha=1.0),
    "LinearRegression": LinearRegression(),
    "RandomForest": RandomForestRegressor(n_estimators=100, random_state=2)
}

results = {}
plt.figure(figsize=(18, 5))

for i, (name, model) in enumerate(models.items(), 1):
    model.fit(x_train, y_train)
    y_pred_val = model.predict(x_val)
    rmse = np.sqrt(mean_squared_error(y_val, y_pred_val))
    results[name] = {
        'rmse': rmse,
        'actual': np.expm1(y_val),
        'predicted': np.expm1(y_pred_val)
    }

    plt.subplot(1, 3, i)
    plt.scatter(results[name]['actual'], results[name]['predicted'], alpha=0.5)
    plt.plot([results[name]['actual'].min(), results[name]['actual'].max()],
             [results[name]['actual'].min(), results[name]['actual'].max()], 'r--')
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
```

```
    plt.title(f'{name}: Actual vs Predicted')

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




```
# Print RMSE Comparison
print("\n--- RMSE Comparison ---")
for name, res in results.items():
    print(f"{name}: RMSE = {res['rmse']:.2f}")
```

```
--- RMSE Comparison ---
Ridge: RMSE = 0.51
LinearRegression: RMSE = 0.51
RandomForest: RMSE = 0.13
```

```
# Bar graph for RMSE comparison
model_names = list(results.keys())
rmse_values = [results[name]['rmse'] for name in model_names]

plt.figure(figsize=(8, 5))
sns.barplot(x=model_names, y=rmse_values, palette='viridis')
plt.ylabel('RMSE')
plt.title('RMSE Comparison of Models')
plt.grid(axis='y')
plt.show()
```

```
<ipython-input-12-9e8b1a11f1f7>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=model_names, y=rmse_values, palette='viridis')
```