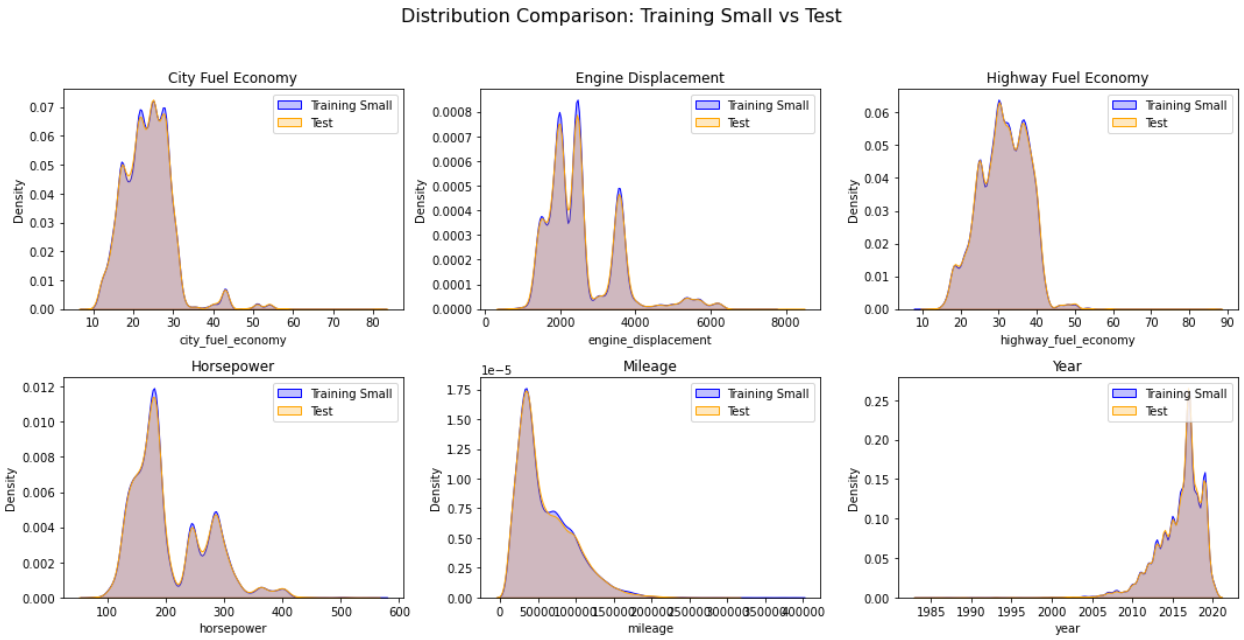
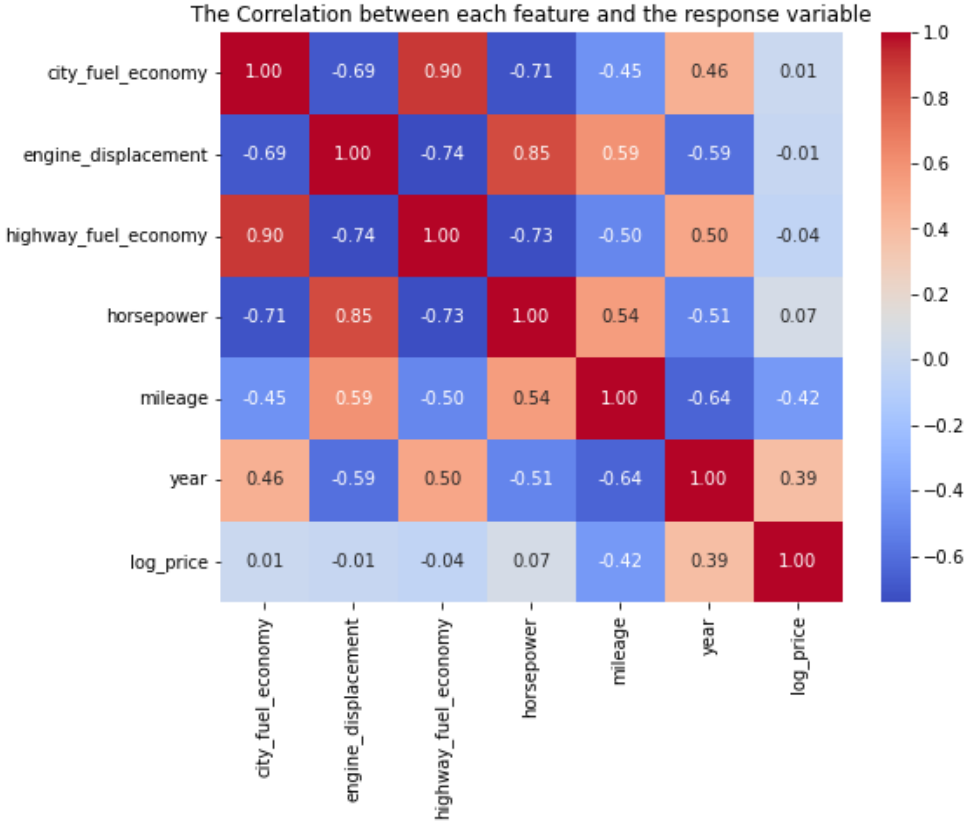
Jand96

MSE: 0.0090 / $R^2 = 0.7234$





I have three training data and one test data, I want to find which training data has similar distruibution to the test data. I want you to make a code that draw a figure that compares the distribution of each feature in my training data vs. in the test set. THe featrues are city_fuel_economy engine_displacement highway_fuel_economy horsepower mileage year
and I want total three graphs.

Certainly! To compare the distributions of each feature across three training datasets and a test dataset, you can use Python's `pandas`, `matplotlib`, and `seaborn` libraries. Below is a step-by-step guide and the corresponding code to help you achieve this visualization.

Overview

1. **Data Preparation**: Ensure your training and test datasets are loaded into pandas DataFrames.
2. **Visualization**: For each training dataset, create a figure with subplots comparing each feature's distribution to that of the test dataset.
3. **Customization**: Use KDE (Kernel Density Estimate) plots for smooth distribution comparison, but you can also opt for histograms or boxplots based on your preference.

Step-by-Step Implementation

```
training_datasets = {
    'Training 1': train1,
    'Training 2': train2,
    'Training 3': train3
}

# Define the plotting function
def plot_distributions(training_name, train_df, test_df, features):
    num_features = len(features)
    cols = 3
    rows = 2
    fig, axes = plt.subplots(rows, cols, figsize=(cols*5, rows*4))
    fig.suptitle(f'Distribution Comparison: {training_name} vs Test', fontsize=16)

    for i, feature in enumerate(features):
        row = i // cols
        col = i % cols
        ax = axes[row, col]

        sns.kdeplot(
            train_df[feature],
            label=training_name,
            shade=True,
            color='blue',
            ax=ax
        )

        sns.kdeplot(
            test_df[feature],
            label='Test',
            shade=True,
            color='orange',
            ax=ax
        )

        ax.set_title(feature.replace('_', ' ').title())
        ax.legend()

    # Remove empty subplots if any
    total_subplots = rows * cols
    if num_features < total_subplots:
        for j in range(num_features, total_subplots):
            fig.delaxes(axes.flatten()[j])

    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.show()

# Generate plots for each training dataset
for name, df in training_datasets.items():
    plot_distributions(name, df, test, features)
```

Interpreting the Plots

Each figure corresponds to one training dataset compared against the test dataset. Within each figure:

- **Subplots**: Each subplot represents a feature, displaying its distribution in both the training and test datasets.
- **Overlayed Distributions:** By overlaying the distributions, you can visually assess how similar or different each training dataset is to the test dataset for each feature.
- **Legends**: Clearly distinguish between training and test distributions to facilitate comparison.

Conclusion

This approach provides a clear visual comparison of feature distributions across multiple training datasets and a test dataset. By examining these plots, you can determine which training dataset(s) closely resemble the test data, aiding in selecting the most appropriate training data for your model.

Feel free to adjust the code to better fit your specific data structures and visualization preferences!