predictor_lr

March 12, 2024

[14]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
```

[15]:

```
def load_dataset(file_path):
    """Load the dataset from a CSV file"""
    return pd.read_csv(file_path)
```

[16]:

```
def preprocess_data(data):
    """Preprocess the dataset by handling missing values"""
    # Fill missing values with the mean of the respective column
    data = data.fillna(data.mean())
    return data
```

[17]:

```
def explore_dataset(data):
    """Display the first few rows and information of the dataset"""
    print("First few rows of the dataset:")
    print(data.head())
    print("\nDataset information:")
    print(data.info())
```

[18]:

```
def visualize_data(data):
    """Visualize the data"""
    # Pairplot to visualize relationships between features
    sns.pairplot(data, x_vars=data.columns[:-1], y_vars=['medv'],␣
 ↪kind='scatter')
    plt.title("Pairplot of Features vs. Target")
    plt.show()

    # Heatmap to visualize correlations between features
    plt.figure(figsize=(12, 8))
    sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title("Correlation Heatmap")
    plt.show()
```

[19]:

```
def split_data(data):
    """Split the dataset into features (X) and target (y)"""
    X = data.drop('medv', axis=1)
    y = data['medv']
    return X, y
```

[20]:

```
def train_model(X_train, y_train):
    """Train a linear regression model"""
    model = LinearRegression()
    model.fit(X_train, y_train)
    return model
```

[21]:

```
def evaluate_model(model, X_test, y_test):
    """Evaluate the trained model"""
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print("Mean Squared Error:", mse)
```

[22]:

```
def visualize_predictions(y_test, y_pred):
    """Visualize the actual vs. predicted prices"""
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, color='blue')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],␣
 ↪linestyle='--', color='red')
    plt.xlabel("Actual Price")
    plt.ylabel("Predicted Price")
    plt.title("Actual vs. Predicted Prices")
    plt.show()
```

[23]:

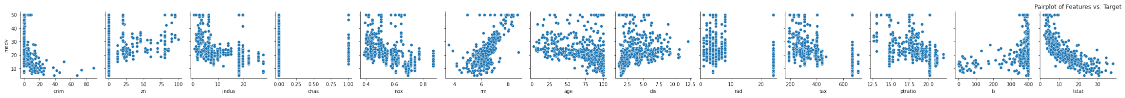
```
def predict_new_house_price(model, new_house_features):
    """Predict the price of a new house"""
    predicted_price = model.predict(new_house_features)
    formatted_price = "${:,.2f}".format(predicted_price[0])
    print("Predicted Price of the House:", formatted_price)
```

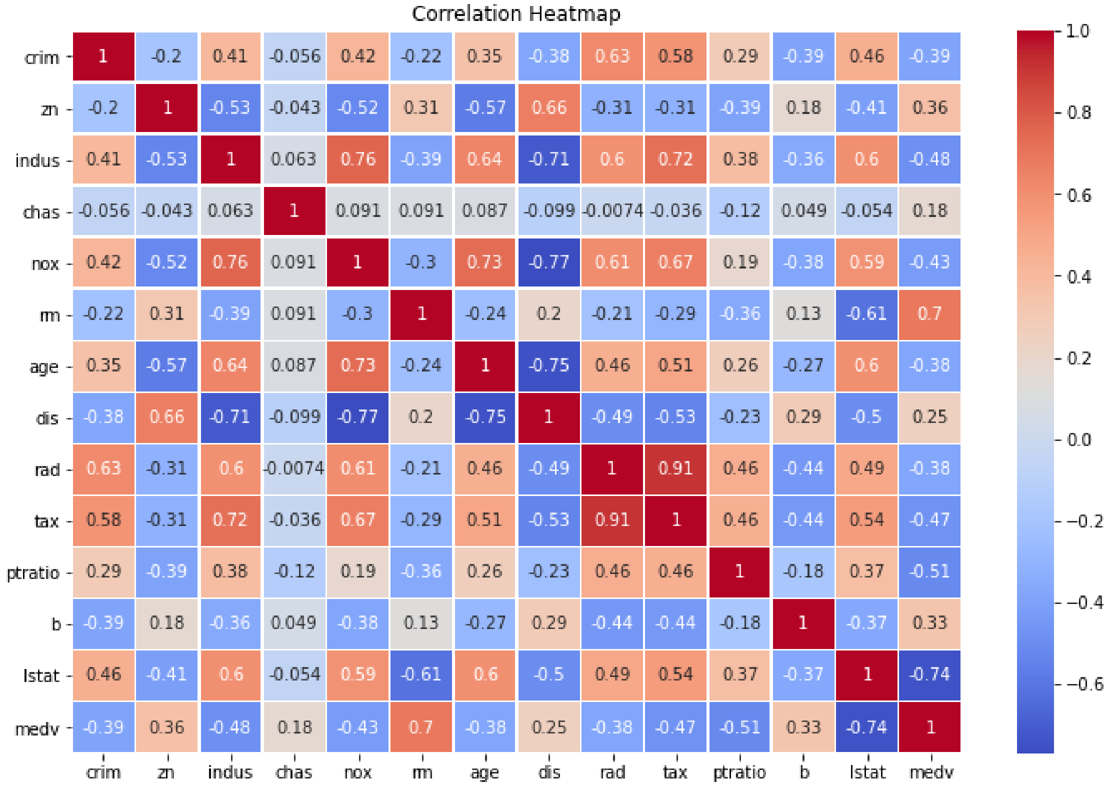
[24]:

```
def main():
    # Load the dataset
    file_path = 'dataset/BostonHousing.csv'
    boston = load_dataset(file_path)

    # Preprocess the dataset
    boston = preprocess_data(boston)

    # Explore the dataset
    explore_dataset(boston)
    visualize_data(boston)
    # Split the data into training and testing sets
    X, y = split_data(boston)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

    # Train the linear regression model
    model = train_model(X_train, y_train)

    # Evaluate the model
    evaluate_model(model, X_test, y_test)

    # Make predictions
    y_pred = model.predict(X_test)
    visualize_predictions(y_test, y_pred)

    # Predict the price of a new house
    new_house_features = np.array([[0.00632, 18, 2.31, 0, 0.538, 6.575, 65.2, 4.
↪09, 1, 296, 15.3, 396.9, 4.98]])
    predict_new_house_price(model, new_house_features)
```

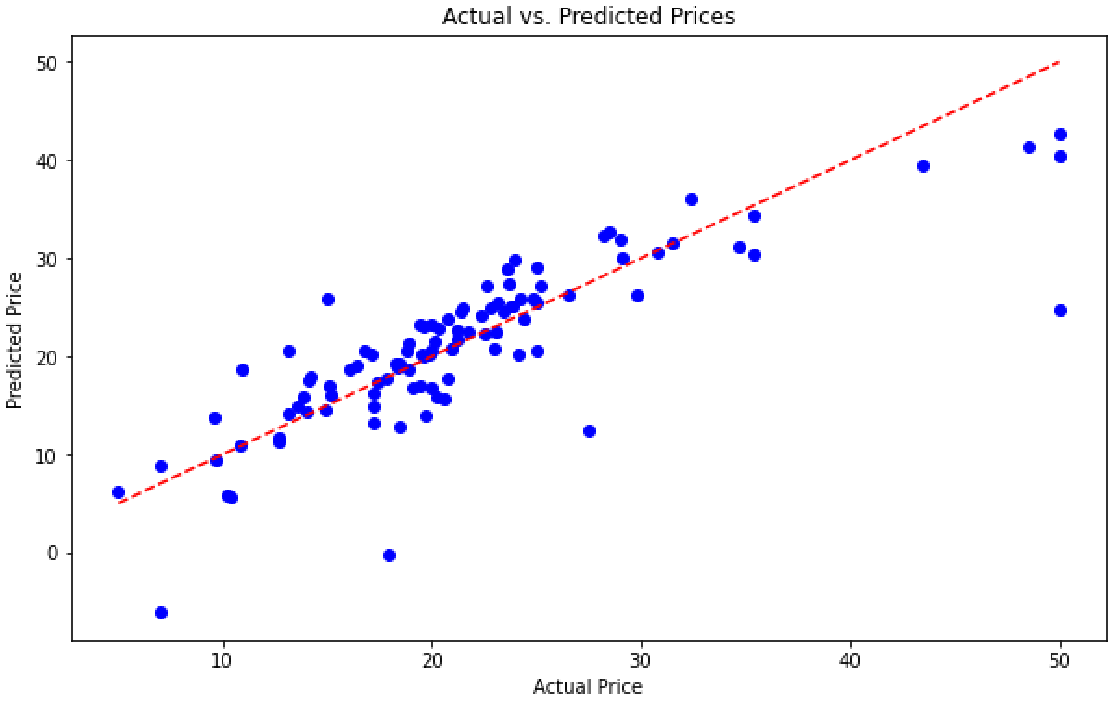
[25]:
```
if __name__ == "__main__":
    # Hide warning concerning feature names
    import warnings
    warnings.filterwarnings("ignore", message="X does not have valid feature␣
↪names, but LinearRegression was fitted with feature names")

    main()
```

```
First few rows of the dataset:
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2

Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
None
```





```
Mean Squared Error: 24.40482518814633
```



```
Predicted Price of the House: $29.94
```

[ ]: