Data imputation is a process of replacing missing or null values in a dataset with statistically estimated values. Here's a step-by-step approach to perform data imputation on the given dataset:

Import necessary libraries

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

Load the dataset

```
# Load the dataset
df = pd.read_csv('your_data.csv')
```

Check for missing values

```
# Check for missing values
print(df.isnull().sum())
```

Impute missing values

We'll use a combination of mean, median, and mode imputation for numerical and categorical columns, respectively. For more complex imputation, we'll use a random forest regressor.

```
# Separate numerical and categorical columns
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Impute missing values in numerical columns
imputer = SimpleImputer(strategy='mean')
df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

# Impute missing values in categorical columns
for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

# Impute missing values in 'price' column using a random forest regressor
X = df.drop(['price'], axis=1)
y = df['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a random forest regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Predict missing values in 'price' column
missing_price_idx = df['price'].isnull()
df.loc[missing_price_idx, 'price'] = rf.predict(X.loc[missing_price_idx])
```

Evaluate the imputation

To evaluate the imputation, we can compare the mean squared error (MSE) of the predicted values with the actual values.

```
# Evaluate the imputation
y_pred = rf.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
```

Note that this is a basic example of data imputation. The choice of imputation strategy depends on the specific characteristics of the dataset and the problem you're trying to solve.

Full Code

Here's the full code with example usage:

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

def load_data(file_path):
    return pd.read_csv(file_path)

def impute_missing_values(df):
    numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
    categorical_cols = df.select_dtypes(include=['object']).columns

    imputer = SimpleImputer(strategy='mean')
    df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

    for col in categorical_cols:
        df[col] = df[col].fillna(df[col].mode()[0])

    X = df.drop(['price'], axis=1)
    y = df['price']

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

    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)

    missing_price_idx = df['price'].isnull()
    df.loc[missing_price_idx, 'price'] = rf.predict(X.loc[missing_price_idx])

    return df

def evaluate_imputation(df):
    X = df.drop(['price'], axis=1)
    y = df['price']

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

    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)

    y_pred = rf.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)

    return mse

def main():
    file_path = 'your_data.csv'
    df = load_data(file_path)
    df = impute_missing_values(df)
    mse = evaluate_imputation(df)
    print(f'Mean Squared Error: {mse:.2f}')
```

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

Replace `'your_data.csv'` with the actual file path of your dataset.