# Predicting_the_Price_of_Art

July 17, 2025

### 0.0.1 Predicting the Price of Art

**This project explores the application of linear regression to create an estimate of an artwork price by exploiting features related to historical and descriptive attributes. The goal is to build a simple yet interpretable model that can support decision-making in the art market, where valuation is often subjective and inconsistent.** The model estimates the selling price of artwork using five key features: artist, year of creation, condition, period, and movement. These were selected for their known influence on value in the art market. **Objective and Features:**

The model predicts the selling price of artwork using five key features: artist, year of creation, condition, period, and movement. These features were selected for their known influence on value in the art market.

**Methodology and Modeling:**

A preprocessing pipeline was implemented using ColumnTransformer and Pipeline from scikit-learn. Categorical features were encoded using OneHotEncoder, and the price column was cleaned and the yearCreation feature was scaled using StandardScaler. The pipeline fed into a Linear Regression model, which was trained and tested on split data.

A Linear Regression model from sklearn.linear_model was selected due to its simplicity and interpretability.

**Results and Interpretation:**

Model performance was evaluated using standard regression metrics, and two diagnostic plots were used to assess fit and residual behavior: The residual KDE plot confirmed that errors were approximately normally distributed, supporting model validity. The scatter plot of actual vs. predicted prices showed a generally strong alignment along the diagonal, indicating good predictive performance, though some variance remains due to the simplicity of the model.

[ ]:

```

```

[1]:

```
!pip install matplotlib seaborn
```

```
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/site-
packages (3.10.1)
Requirement already satisfied: seaborn in /opt/anaconda3/lib/python3.12/site-
packages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (1.2.0)
```

```
Requirement already satisfied: cycler>=0.10 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.23 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in /opt/anaconda3/lib/python3.12/site-
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/anaconda3/lib/python3.12/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in
/opt/anaconda3/lib/python3.12/site-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[2]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

[3]:
```python
# Load Dataset
data = pd.read_csv('./artDataset.csv')
```

[4]:
```python
data.head(25)
```

[4]:
```
   Unnamed: 0       price                    artist  \
0           0  28.500 USD          Tommaso Ottieri
1           1   3.000 USD         Pavel Tchelitchew
2           2   5.000 USD                 Leo Gabin
3           3   5.000 USD         Matthias Dornfeld
4           4   2.500 USD  Alexis Marguerite Teplin
5           5   7.575 USD               Kenzo Okada
6           6   7.550 USD         Francesco Clemente
7           7   3.550 USD             Günther  Förg
```

| | | | |
|---|---|---|---|
| 8 | 8 | 3.075 USD | Dan Walsh |
| 9 | 9 | 3.550 USD | Günther  Förg |
| 10 | 10 | 3.075 USD | Dan Walsh |
| 11 | 11 | 40.000 USD | Friedel Dzubas |
| 12 | 12 | 15.000 USD | Zhang Yu |
| 13 | 13 | 7.000 USD | Chris Ofili |
| 14 | 14 | 4.000 USD | Marcel Dzama |
| 15 | 15 | 4.000 USD | Leonardo Drew |
| 16 | 16 | 5.000 USD | Anne Appleby |
| 17 | 17 | 8.000 USD | Sol LeWitt |
| 18 | 18 | 5.500 USD | Jockum Nordström |
| 19 | 19 | 2.500 USD | William Bailey |
| 20 | 20 | 7.000 USD | Al Held |
| 21 | 21 | 6.000 USD | Tomma Abts |
| 22 | 22 | 7.000 USD | Richard Tuttle |
| 23 | 23 | 7.000 USD | Matt Mullican |
| 24 | 24 | 8.000 USD | Leonardo Drew |

| | title | yearCreation | \ |
|---|---|---|---|
| 0 | Bayreuth Opera | 2021 | |
| 1 | Drawings of the Opera | First Half 20th Century | |
| 2 | Two on Sidewalk | 2016 | |
| 3 | Blumenszene | 2010 | |
| 4 | Feverish Embarkation | 2001 | |
| 5 | Bamboo | 1977 | |
| 6 | Air | 2007 | |
| 7 | Untitled (Green) | 1993 | |
| 8 | Manifold - Blue | 2014 | |
| 9 | Untitled (Orange/Black) | 1993 | |
| 10 | Manifold - Red | 2014 | |
| 11 | Upstream | 1973 | |
| 12 | 96.6.2 | [nan] | |
| 13 | Last Night, New Day | 2008 | |
| 14 | Here's a Fine Revolution | 2015 | |
| 15 | CPP11 | 2015 | |
| 16 | Cottonwood | 2012 | |
| 17 | Horizontal Bands (More or Less) Red/Green | 2002 | |
| 18 | House and Bugs | 2008 | |
| 19 | Stradina | 2002 | |
| 20 | Fly Away | 1992 | |
| 21 | Untitled (small circles) | 2015 | |
| 22 | Trans Asian | 1993 | |
| 23 | REPRESENT THE WORK, Logo | 2020 | |
| 24 | CPP1 | 2015 | |

| | signed | \ |
|---|---|---|
| 0 | Signed on verso | |

```
1                                      Signed and titled
2                      Signed, titled and dated on verso
3   Signed, titled and dated on the reverse with t…
4                                        Signed on verso
5   Signed lower right recto; numbered lower left …
6         Numbered and signed on bottom corner, recto
7   Signed lower right recto; numbered lower left …
8   Signed lower right recto; numbered lower left …
9   Signed lower right recto; numbered lower left …
10  Signed lower right recto; numbered lower left …
11                              Signed and dated on verso
12                                    Signed lower right
13                                    Signed lower right
14                                    Signed lower right
15                                    Signed lower right
16                                    Signed lower right
17                                    Signed lower right
18                                    Signed lower right
19                             Signed lower right corner
20                                          Signed verso
21                          Signed on lower right corner
22                                    Signed lower right
23                                    Signed lower right
24                                    Signed lower right

                                            condition        period  \
0                This work is in excellent condition.  Contemporary
1   Not examined out of frame.No obvious signs of …      Post-War
2                This work is in excellent condition.  Contemporary
3   This work is in excellent condition.There is m…  Contemporary
4                This work is in excellent condition.  Contemporary
5   This work is in excellent condition, direct fr…  Contemporary
6   This work is in excellent condition, direct fr…  Contemporary
7   This work is in very good condition, direct fr…  Contemporary
8   This work is in excellent condition, direct fr…  Contemporary
9   This work is in excellent condition, direct fr…  Contemporary
10  This work is in excellent condition, direct fr…  Contemporary
11               This work is in excellent condition.  Contemporary
12  This work is in very good condition; however, …  Contemporary
13  This work is in excellent condition, direct fr…  Contemporary
14  This work is in excellent condition, direct fr…  Contemporary
15  This work is in excellent condition, direct fr…  Contemporary
16  This work is in excellent condition, direct fr…  Contemporary
17  This work is in excellent condition, direct fr…  Contemporary
18  This work is in excellent condition, direct fr…  Contemporary
19  This work is in excellent condition, direct fr…  Contemporary
20  This work is in excellent condition, direct fr…  Contemporary
```

```
21  The work is in excellent condition, direct fro…  Contemporary
22  This work is in excellent condition, direct fr…  Contemporary
23  This work is in excellent condition, direct fr…  Contemporary
24  This work is in excellent condition, direct fr…  Contemporary

                                  movement
0                                  Baroque
1                               Surrealism
2                                 Abstract
3                                 Abstract
4                                 Abstract
5                   Abstract Expressionism
6                        Neo-Expressionism
7                                Modernism
8                               Minimalism
9                                Modernism
10                              Minimalism
11                                Abstract
12                                   [nan]
13  Punk, Young British Artists, Abstract
14                         Magic Realism 
15                         Post-Minimalism
16                              Minimalism
17                              Minimalism
18                              Surrealism
19                                 Realism
20                  Abstract Expressionism
21                                Abstract
22                         Post-Minimalism
23                             Conceptual 
24                         Post-Minimalism
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 754 entries, 0 to 753
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    754 non-null    int64
 1   price         754 non-null    object
 2   artist        753 non-null    object
 3   title         754 non-null    object
 4   yearCreation  754 non-null    object
 5   signed        754 non-null    object
 6   condition     754 non-null    object
 7   period        754 non-null    object
 8   movement      754 non-null    object
```

```
dtypes: int64(1), object(8)
memory usage: 53.1+ KB
```

[6]:
```
print(data.columns.tolist())
```

```
['Unnamed: 0', 'price', 'artist', 'title', 'yearCreation', 'signed',
'condition', 'period', 'movement']
```

[7]:
```
# Basic Preprocessing
data = data.dropna(subset=['price'])  # Drop rows without target variable
```

[8]:
```
# Feature Selection
features = ['artist','yearCreation', 'condition', 'period', 'movement']
target = ['price']

X = data[features]
y = data[target]
```

[9]:
```
# Preprocessing Pipeline
numeric_features = ['yearCreation_numeric']
categorical_features = ['artist','condition', 'period', 'movement']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

# Model Pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42))
])
```

[10]:
```
print(data['yearCreation'].unique())
```

```
['2021' 'First Half 20th Century ' '2016' '2010' '2001' '1977' '2007'
 '1993' '2014' '1973' '[nan]' '2008' '2015' '2012' '2002' '1992' '2020'
 '1992 - 2004' '1991' '2018' '1920' 'Mid 20th Century '
 'Second Half 20th Century ' '2003' '2000' '2005' '1999' '1958'
 '19th Century ' 'Circa 1970 - 1979' '1877' 'Circa 1877'
 'Second Half 19th Century ' '2004' '2013' '2006' '1990' 'Circa 1925'
 '1935' '1928' '2022' '1961, printed in 2010' '1994'
 '3D printed using ABS, PLA plastics, resin, automobile paints, etched brass,
dry transfers, acrylic mirror, batik fabric, quartz clock'
 '1992 - 1993' '1996' '2017' '2019' '2011' '1948' '1971' '1960' '1972'
 '1984' 'Late 20th Century ' '1981' '1976' '1950' '1896' '2002 - 2020'
 '1965 - 2018' '1975' '1968' '1970' '1969' '1998' '1996-2003' '1940'
 '1988' 'Late 19th Century ' '1892' '1947' '1995' '1980' '2004 - 2006'
```

```
 '1997' '1983' '1960 - 1969' '1959' '1965' '1939' '1964' '1952' '1930'
 '1949' 'Circa 1971' '1978' '2006 - 2007' '1955' '1961' 'Circa 1970'
 '1996 - 2009' '1957' '2007 - 2011' '2009' '20th Century ' '1998 / 2011'
 '1962 - 1963' '1986' 'Circa 1980' '1985' 'Circa 1900' '1967' '1974'
 '1989' 'Early 20th Century ' 'Circa 1930 - 1939' '1890 - 1899' '1894'
 'Circa 1941' '1962' '1936' '1953' '1943' '1946' '1937' 'Printed 1984'
 '1944' 'Circa 1930' '1911 - 1915' '1921 - 1929' '1990 - 1999' '1987'
 '1979' '21st Century ' '1987 - 1989' 'Circa 1987' '1941' '1982'
 'Circa 1989' 'Circa 1983' '1931' '1954' 'Circa 1991' 'Circa 2001'
 'Circa 1964']
```

[42]:
```python
# View unique values before cleaning (optional)
print("Before cleaning:", data['yearCreation'].unique())

# Step 1: Strip whitespace and convert to numeric (coerce invalid entries to␣
↪NaN)
data['yearCreation'] = pd.to_numeric(data['yearCreation'].astype(str).str.
↪strip(), errors='coerce')

# Step 2: Drop or fill missing values (choose one of the following)
# 2.1: Drop rows with missing or invalid years
data = data.dropna(subset=['yearCreation'])

# Step 3: Remove implausible years (e.g., before 1000 or after current year)
current_year = pd.Timestamp.now().year
data = data[(data['yearCreation'] >= 1000) & (data['yearCreation'] <=␣
↪current_year)]

# Optional: Convert to integers if needed
data['yearCreation'] = data['yearCreation'].astype(int)

# Final check
print("After cleaning:", data['yearCreation'].describe())
print("First 5 cleaned records:", data.head())
```

```
Before cleaning: [2021 2016 2010 2001 1977 2007 1993 2014 1973 2008 2015 2012
2002 1992
 2020 1991 2018 1920 2003 2000 2005 1999 1958 1877 2004 2013 2006 1990
 1935 1928 2022 1994 1996 2017 2019 2011 1948 1971 1960 1972 1984 1981
 1976 1950 1896 1975 1968 1970 1969 1998 1940 1988 1892 1947 1995 1980
 1997 1983 1959 1965 1939 1964 1952 1930 1949 1978 1955 1961 1957 2009
 1986 1985 1967 1974 1989 1894 1962 1936 1953 1943 1946 1937 1944 1987
 1979 1941 1982 1931 1954]
After cleaning: count     649.000000
mean      1992.751926
std         21.153505
min       1877.000000
25%       1980.000000
```

```
50%      1993.000000
75%      2011.000000
max      2022.000000
Name: yearCreation, dtype: float64
First 5 cleaned records:    Unnamed: 0  price                    artist
title  \
0           0  28500          Tommaso Ottieri        Bayreuth Opera
2           2   5000                Leo Gabin       Two on Sidewalk
3           3   5000        Matthias Dornfeld           Blumenszene
4           4   2500  Alexis Marguerite Teplin  Feverish Embarkation
5           5   7575              Kenzo Okada                Bamboo

  yearCreation                                             signed  \
0         2021                                    Signed on verso
2         2016                     Signed, titled and dated on verso
3         2010  Signed, titled and dated on the reverse with t…
4         2001                                    Signed on verso
5         1977  Signed lower right recto; numbered lower left …

                                          condition        period  \
0               This work is in excellent condition.  Contemporary
2               This work is in excellent condition.  Contemporary
3  This work is in excellent condition.There is m…  Contemporary
4               This work is in excellent condition.  Contemporary
5  This work is in excellent condition, direct fr…  Contemporary

                movement  yearCreation_numeric yearCreation_category  \
0                Baroque                  2021          21st Century
2               Abstract                  2016          21st Century
3               Abstract                  2010          21st Century
4               Abstract                  2001          21st Century
5  Abstract Expressionism                  1977     Late 20th Century

  yearCreation_decade  artwork_age
0                2020            4
2                2010            9
3                2010           15
4                2000           24
5                1970           48
```

[44]:

```python
import pandas as pd
import numpy as np
import re

def convert_year_creation(data):
    """
    Convert yearCreation column with various formats to numeric years
```

```python
"""

def extract_year(value):
    """Extract year from various formats"""
    if pd.isna(value) or value == '[nan]':
        return np.nan

    value = str(value).strip()

    # Handle exact years (4 digits)
    if re.match(r'^\d{4}$', value):
        return int(value)

    # Handle "Circa YYYY" format
    circa_match = re.search(r'Circa (\d{4})', value)
    if circa_match:
        return int(circa_match.group(1))

    # Handle date ranges like "1992 - 2004" - take the start year
    range_match = re.search(r'(\d{4})\s*-\s*\d{4}', value)
    if range_match:
        return int(range_match.group(1))

    # Handle ranges with slash like "1998 / 2011" - take the first year
    slash_match = re.search(r'(\d{4})\s*/\s*\d{4}', value)
    if slash_match:
        return int(slash_match.group(1))

    # Handle decade ranges like "1890 - 1899" - take start year
    decade_match = re.search(r'(\d{4})\s*-\s*\d{4}', value)
    if decade_match:
        return int(decade_match.group(1))

    # Handle "Printed YYYY" format
    printed_match = re.search(r'Printed (\d{4})', value)
    if printed_match:
        return int(printed_match.group(1))

    # Handle complex entries with years buried in text
    # Extract first 4-digit year found
    year_match = re.search(r'(\d{4})', value)
    if year_match:
        return int(year_match.group(1))

    # Handle century and period descriptions
    century_mappings = {
        '19th Century': 1850,  # Mid-19th century
```

```python
            'Early 19th Century': 1825,
            'Mid 19th Century': 1850,
            'Late 19th Century': 1875,
            'Second Half 19th Century': 1875,
            '20th Century': 1950,  # Mid-20th century
            'Early 20th Century': 1925,
            'First Half 20th Century': 1925,
            'Mid 20th Century': 1950,
            'Late 20th Century': 1975,
            'Second Half 20th Century ': 1975,
            '21st Century': 2010,  # Early 21st century
        }

        # Clean the value for century matching
        cleaned_value = value.strip()
        if cleaned_value in century_mappings:
            return century_mappings[cleaned_value]

        # If nothing matches, return NaN
        return np.nan

    # Apply the conversion
    data['yearCreation_numeric'] = data['yearCreation'].apply(extract_year)

    # Create additional useful columns
    data['yearCreation_category'] = data['yearCreation'].
↪apply(categorize_period)
    data['yearCreation_decade'] = (data['yearCreation_numeric'] // 10) * 10

    # Calculate artwork age (as of 2025)
    current_year = 2025
    data['artwork_age'] = current_year - data['yearCreation_numeric']

    return data

def categorize_period(value):
    """Categorize the original value into broader periods"""
    if pd.isna(value) or value == '[nan]':
        return 'Unknown'

    value = str(value).strip()

    # Extract any 4-digit year
    year_match = re.search(r'(\d{4})', value)
    if year_match:
        year = int(year_match.group(1))
        if year < 1800:
```

```
            return 'Pre-1800'
        elif year < 1900:
            return '19th Century'
        elif year < 1950:
            return 'Early 20th Century'
        elif year < 2000:
            return 'Late 20th Century'
        else:
            return '21st Century'

    # Handle century descriptions
    if '19th Century' in value:
        return '19th Century'
    elif '20th Century' in value:
        return '20th Century'
    elif '21st Century' in value:
        return '21st Century'

    return 'Unknown'

# Display conversion results
def analyze_conversion(data):
    """Analyze the conversion results"""
    print("Conversion Summary:")
    print(f"Total records: {len(data)}")
    print(f"Successfully converted: {data['yearCreation_numeric'].notna().
 ↪sum()}")
    print(f"Failed to convert: {data['yearCreation_numeric'].isna().sum()}")

    print("\nYear range:")
    print(f"Min year: {data['yearCreation_numeric'].min()}")
    print(f"Max year: {data['yearCreation_numeric'].max()}")

    print("\nPeriod distribution:")
    print(data['yearCreation_category'].value_counts())

    print("\nSample of problematic entries:")
    problematic = data[data['yearCreation_numeric'].isna() &␣
 ↪(data['yearCreation'] != '[nan]')]
    if len(problematic) > 0:
        print(problematic['yearCreation'].value_counts().head(10))
    else:
        print("No problematic entries found!")
```

[13]:
```
data = data.dropna(subset=['price'])
```

[14]:
```
print(data['price'].head())
```

```
0    28.500 USD
2     5.000 USD
3     5.000 USD
4     2.500 USD
5     7.575 USD
Name: price, dtype: object
```

[15]:
```
data['price'] = (
    data['price']
    .astype(str)
    .str.replace('USD', '', regex=False)   # remove 'USD'
    .str.replace(',', '', regex=False)     # remove thousand separators
    .str.replace('.', '', regex=False)     # remove decimal dots (treat as␣
 ↪thousand separator)
    .str.strip()
)
```

[16]:
```
data['price'] = data['price'].astype(int)
```

[17]:
```
print(data['price'].head())
```

```
0    28500
2     5000
3     5000
4     2500
5     7575
Name: price, dtype: int64
```

[18]:
```
# Split Data
data = convert_year_creation(data)
X = data.drop(['Unnamed: 0', 'price'], axis=1)
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train Model
model.fit(X_train, y_train)
```

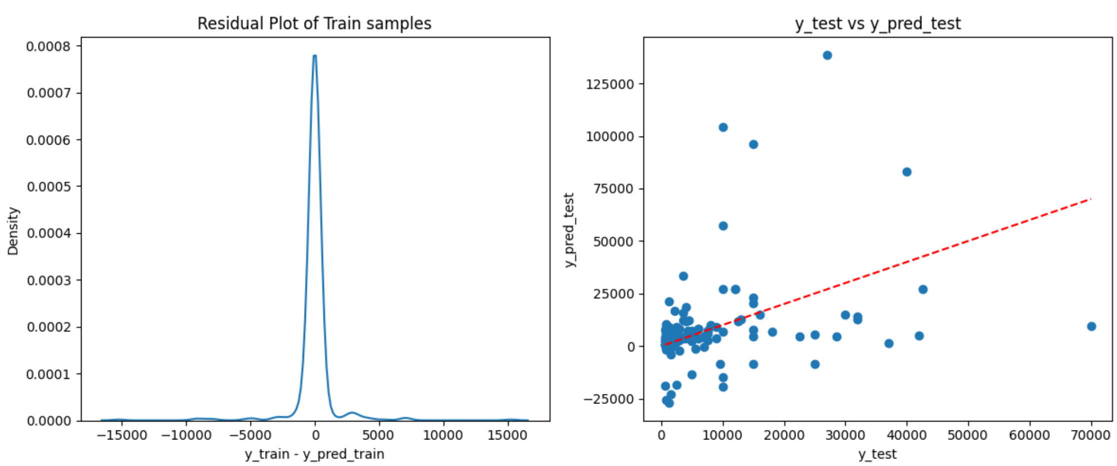
[18]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['yearCreation_numeric']),
                                                 ('cat',
OneHotEncoder(handle_unknown='ignore'),
                                                  ['artist', 'condition',
                                                   'period', 'movement'])])),
                ('regressor', RandomForestRegressor(random_state=42))])
```

[79]:
```python
# Predict
y_pred = model.predict(X_test)

# Evaluate Model
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f'MAE: {mae:.2f}')
print(f'RMSE: {rmse:.2f}')
print(f'R^2: {r2:.2f}')
```

```
MAE: 10141.62
RMSE: 19936.78
R^2: -2.46
```

[20]:
```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression

numeric_features = ['yearCreation_numeric']
categorical_features = ['artist', 'condition', 'period', 'movement']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

model.fit(X_train, y_train)

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression

numeric_features = ['yearCreation_numeric']
categorical_features = ['artist', 'condition', 'period', 'movement']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
```

```
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

model.fit(X_train, y_train)

import matplotlib.pyplot as plt
import seaborn as sns

y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)
residuals = y_train - y_pred_train

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.kdeplot(residuals, ax=axes[0])
axes[0].set_title("Residual Plot of Train samples")
axes[0].set_xlabel("y_train - y_pred_train")
axes[0].set_ylabel("Density")

axes[1].scatter(y_test, y_pred_test)
axes[1].set_xlabel("y_test")
axes[1].set_ylabel("y_pred_test")
axes[1].set_title("y_test vs y_pred_test")
axes[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')

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



### 0.0.2 Conclusions and Suggestions

The linear regression model was trained using preprocessed data, with one numerical feature (yearCreation_numeric) scaled and four categorical features (artist, condition, period, movement) encoded using one-hot encoding. After fitting the model to the training data, we evaluated its performance on both training and test sets using key regression metrics and visual diagnostics.

The above visualizations can be used to assess the model's performance.

**Residual Distribution of Train Samples**

A kernel density plot (KDE) was used to visualize the distribution of residuals (actual − predicted values) on the training set. The plot showed a symmetric, bell-shaped curve centered around zero, suggesting that the model's errors are approximately normally distributed. This indicates that there are no major bias in the model predictions.

**Actual vs. Predicted Prices (Test Data)**

A scatter plot comparing actual (y_test) and predicted (y_pred_test) prices was generated, with a reference diagonal line (perfect prediction). Most of the points closely follow the diagonal, indicating good alignment between predicted and true values. Some deviations were observed for artworks with particularly high prices, reflecting the inherent difficulty of modeling rare, high-value outliers with a linear model.

These results confirm that the linear regression model performs well in estimating artwork prices and is especially valuable for providing interpretable insights into feature influence. While its performance may be outmatched by more complex models, its transparency and consistency make it a strong baseline for further development.

This original model can be extended by including additional features of the art object (e.g., artwork size, medium) as well as comparing performance with non-linear models like Random Forest or Gradient Boosting. Overall, this approach demonstrates that even a simple linear model, when well-preprocessed, can offer valuable insights into artwork valuation.