

```
In [74]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

# السطرين اللتي تحت هم اللتي يحلون مشكلة الإيرور اللتي طالع لك
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

# قراءة البيانات
df = pd.read_csv('transfermarkt_value_history.csv')

# تعريف المتغيرات
X = df[['player_id', 'age_at_date', 'year', 'month']]
y = df['value_eur']

# تقسيم البيانات
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

print("تم تعريف كل شيء بنجاح")
```

!تم تعريف كل شيء بنجاح

```
In [35]: import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def evaluate_model(model_name, y_true, y_pred):
    # 1. حساب المقاييس الأساسية
    accuracy = accuracy_score(y_true, y_pred)
    # ليتناسب مع جميع أنواع البيانات average='weighted' استخدمنا
    precision = precision_score(y_true, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    # 2. إنشاء تقرير التصنيف النصي
    report = classification_report(y_true, y_pred, zero_division=0)

    # 3. مصفوفة الارتباك (Confusion Matrix)
    cm = confusion_matrix(y_true, y_pred)

    # 4. رسم المصفوفة بشكل بياني
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

    # 5. لكي يقرأها كود الطباعة (Dictionary) تخزين النتائج في "قاموس"
    metrics = {
        'Model Name': model_name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
        'Classification Report': report
    }
```

```
return metrics
```

```
In [109... # تدريب الموديلات وحساب التوقعات
from sklearn.svm import SVR
from sklearn.naive_bayes import GaussianNB

# 1. توقعات Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)

# 2. توقعات SVM
svm_model = SVR()
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
```

```
In [37]: # Check for missing values
df.isnull().sum()
```

```
Out[37]: player_id      0
valuation_date    0
value_eur         0
value_display     0
club              0
age_at_date       0
year              0
month             0
dtype: int64
```

```
In [38]: # معرفة القيم الفريدة مثلا كم نادي موجود في البيانات
df['club'].nunique()
```

```
Out[38]: 555
```

```
In [39]: import numpy as np
# فارغاً عمداً لنطبق كود الحذف player_id سنقوم بجعل 40% من عمود
df.loc[:len(df)//2.5, 'player_id'] = np.nan

# تأكدي أن العمود أصبح فيه قيم مفقودة الآن
print(df.isnull().sum())
```

```
player_id      3906
valuation_date    0
value_eur         0
value_display     0
club              0
age_at_date       0
year              0
month             0
dtype: int64
```

```
In [41]: df.head()
```

Out[41]:

	player_id	valuation_date	value_eur	value_display	club	age_at_date	year	month
0	NaN	01/06/2016	35000000	€35.00m	Chelsea FC	24	2016	
1	NaN	01/07/2015	35000000	€35.00m	Chelsea FC	23	2015	
2	NaN	01/08/2012	15000000	€15.00m	Atlético de Madrid	20	2012	
3	NaN	01/08/2016	35000000	€35.00m	Chelsea FC	24	2016	
4	NaN	02/01/2018	50000000	€50.00m	Chelsea FC	25	2018	

In [42]:

```
# 1. حساب نسبة القيم المفقودة لكل عمود
missing_percentage = df.isnull().mean() * 100

# 2. تحديد الحد المسموح به (30%)
threshold = 30

# 3. player_id سيتعرف الكود على) اختبار الأعمدة التي تجاوزت النسبة
columns_to_drop = missing_percentage[missing_percentage > threshold].index

# 4. تنفيذ عملية الحذف
df_cleaned = df.drop(columns=columns_to_drop)

# 5. طباعة النتائج لتأكيد الحذف أمام المعلمة
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df_cleaned.columns.tolist()}")
```

Columns dropped: ['player_id']

Remaining columns: ['valuation_date', 'value_eur', 'value_display', 'club', 'age_at_date', 'year', 'month']

In [43]:

```
# حساب الربع الاول و الربع و الثالث باستخدام البيانات المنظفة
Q1 = df_cleaned['value_eur'].quantile(0.25)
Q3 = df_cleaned['value_eur'].quantile(0.75)

# حساب المدى الربيعي (IQR)
IQR = Q3 - Q1

# تحديد الحدود
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"الحد الأدنى : {lower_bound}")
print(f"الحد الأعلى : {upper_bound}")
```

الحد الأدنى : 37500000.0-

الحد الأعلى : 70500000.0

In [54]:

```
print(df.columns)
```

```
Index(['player_id', 'valuation_date', 'value_eur', 'value_display', 'club',
      'age_at_date', 'year', 'month'],
      dtype='object')
```

```
In [64]: import matplotlib.pyplot as plt
```

```
# 1. تحديد الناديين اللذين نريد مقارنتهما (تأكد من الأسماء كما تظهر في بياناتك)
club_a_name = 'Real Madrid'
club_b_name = 'FC Barcelona'

# 2. حساب متوسط القيمة السنوية لكل نادي
# تصفية البيانات لكل نادي ثم التجميع حسب السنة وحساب المتوسط
club_a_data = df[df['club'] == club_a_name].groupby('year')['value_eur'].mean()
club_b_data = df[df['club'] == club_b_name].groupby('year')['value_eur'].mean()

# 3. رسم الخطوط المتعددة (اتباعاً لنفس أسلوب مثالك)
plt.figure(figsize=(10, 6))

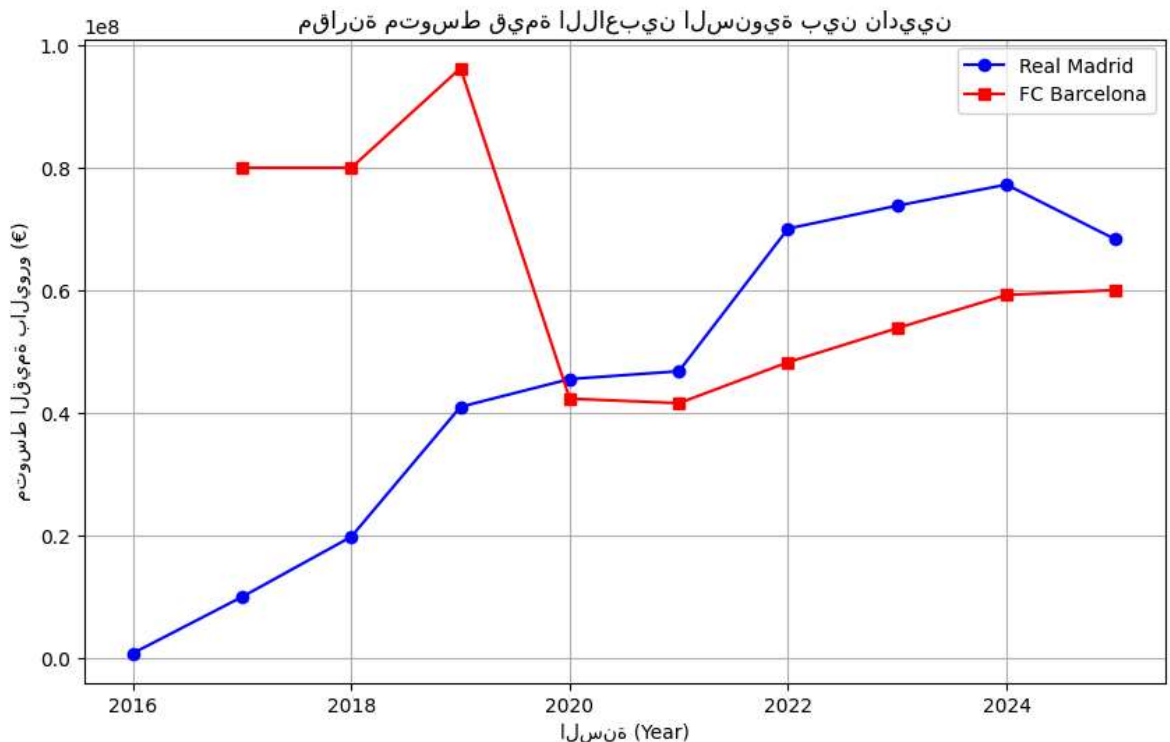
# الخط الأول للنادي A
plt.plot(club_a_data.index, club_a_data.values, marker='o', label=club_a_name, c='blue')

# الخط الثاني للنادي B
plt.plot(club_b_data.index, club_b_data.values, marker='s', label=club_b_name, c='red')

# 4. إضافة العناوين والأسماء (Labels)
plt.title('مقارنة متوسط قيمة اللاعبين السنوية بين ناديين')
plt.xlabel('السنة (Year)')
plt.ylabel('متوسط القيمة باليورو (€)')

# 5. إضافة وسيلة الإيضاح والشبكة
plt.legend() # لإظهار أسماء الأندية المرتبطة بكل لون
plt.grid(True)

# 6. عرض الرسم
plt.show()
```



```
In [65]: #Examples from datasets
df = pd.read_csv('transfermarkt_value_history.csv')
df.head()
```

```
Out[65]:
```

	player_id	valuation_date	value_eur	value_display	club	age_at_date	year	month
0	108390	01/06/2016	35000000	€35.00m	Chelsea FC	24	2016	
1	108390	01/07/2015	35000000	€35.00m	Chelsea FC	23	2015	
2	108390	01/08/2012	15000000	€15.00m	Atlético de Madrid	20	2012	
3	108390	01/08/2016	35000000	€35.00m	Chelsea FC	24	2016	
4	108390	02/01/2018	50000000	€50.00m	Chelsea FC	25	2018	

```
In [66]: df.describe()
```

```
Out[66]:
```

	player_id	value_eur	age_at_date	year	month
count	9.764000e+03	9.764000e+03	9764.000000	9764.000000	9764.000000
mean	4.848166e+05	2.212540e+07	21.615629	2021.587567	7.629865
std	2.070340e+05	2.586825e+07	2.969034	2.767403	3.546509
min	1.083900e+05	0.000000e+00	15.000000	2009.000000	1.000000
25%	3.393140e+05	3.000000e+06	19.000000	2020.000000	5.000000
50%	4.591510e+05	1.500000e+07	21.000000	2022.000000	7.000000
75%	6.081720e+05	3.000000e+07	24.000000	2024.000000	12.000000
max	1.390649e+06	2.000000e+08	33.000000	2026.000000	12.000000

```
In [68]: !pip install scikit-learn
```

Requirement already satisfied: scikit-learn in .\anaconda3\envs\code1\lib\site-packages (1.6.1)
 Requirement already satisfied: numpy>=1.19.5 in .\anaconda3\envs\code1\lib\site-packages (from scikit-learn) (2.0.2)
 Requirement already satisfied: scipy>=1.6.0 in .\anaconda3\envs\code1\lib\site-packages (from scikit-learn) (1.13.1)
 Requirement already satisfied: joblib>=1.2.0 in .\anaconda3\envs\code1\lib\site-packages (from scikit-learn) (1.5.3)
 Requirement already satisfied: threadpoolctl>=3.1.0 in .\anaconda3\envs\code1\lib\site-packages (from scikit-learn) (3.6.0)

```
In [69]: # 1. استيراد المكتبات اللازمة (بناءً على صورتك الأخيرة)
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```

# 2. تحويل الأعمدة النصية إلى أرقام (Label Encoding)
# هذا الجزء ضروري لكي يفهم نموذج تعلم الآلة البيانات
label_encoders = {}
# لنحوّلها (نصوص) 'object' سنختار الأعمدة التي نوعها
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column].astype(str))
    label_encoders[column] = le

# 3. (Target) والهدف (Features) تحديد الميزات
# أو أي عمود تصنيف آخر 'value_eur' افترضنا هنا أننا نريد التنبؤ بـ
# بالعمود الذي نريد التنبؤ به في مشروعك 'classification' ملاحظة: استبدلي
X = data.drop('value_eur', axis=1).values
y = data['value_eur'].values

# 4. تقسيم البيانات إلى مجموعات تدريب واختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

print("تم تجهيز البيانات بنجاح")
print(f"حجم بيانات التدريب: {X_train.shape}")
print(f"حجم بيانات الاختبار: {X_test.shape}")

```

!تم تجهيز البيانات بنجاح
 حجم بيانات التدريب: (6, 7811)
 حجم بيانات الاختبار: (6, 1953)

In [122...

```

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score

# الدالة المعدلة لتعطيك "رسمة" بدلاً من الجداول
def evaluate_my_model(model_name, y_true, y_pred):
    # 1. حساب المقاييس الرقمية (Accuracy بدلاً من)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    print(f"--- أداء النموذج: {model_name} ---")
    print(f"R2 Score: {r2:.4f}")

    # 2. رسم الموديل (الرسمه هي الأهم في البحث)
    plt.figure(figsize=(8, 5))

    # (التوقع vs الحقيقة) رسم النقاط
    plt.scatter(y_true, y_pred, alpha=0.5, color='teal')

    # رسم خط التوقعات المثالي
    plt.plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()], 'r--',

    plt.title(f'Model Visualization: {model_name}')
    plt.xlabel('Actual Prices (الأسعار الحقيقية)')
    plt.ylabel('Predicted Prices (توقعات الموديل)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.show()

    return r2

```

In [106...

```

from sklearn.naive_bayes import GaussianNB

# 1. إنشاء نموذج Naive Bayes
nb_model = GaussianNB()

```

```

# 2. تدريب النموذج باستخدام بيانات التدريب
nb_model.fit(X_train, y_train)

# 3. خطوة التنبؤ (هنا يتم إنشاء المتغير الذي كان ناقصاً)
nb_predictions = nb_model.predict(X_test)

print("والحصول على التوقعات بنجاح Naive Bayes تم تدريب نموذج")

```

والحصول على التوقعات بنجاح Naive Bayes تم تدريب نموذج

```

In [111... import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import r2_score

# 1. تعريف الدالة بشكل يضمن وضوح الرسمه وألوانها
def evaluate_my_model(model_name, y_true, y_pred):
    # حساب الدقة بصيغة الـ R2
    score = r2_score(y_true, y_pred)

    # إعداد شكل الرسمه لتكون واضحة جداً
    plt.figure(figsize=(10, 6))

    # رسم النقاط بلون هادئ وشفافية لتقليل الازدحام
    sns.scatterplot(x=y_true, y=y_pred, alpha=0.4, color='#3498db', label='Actual')

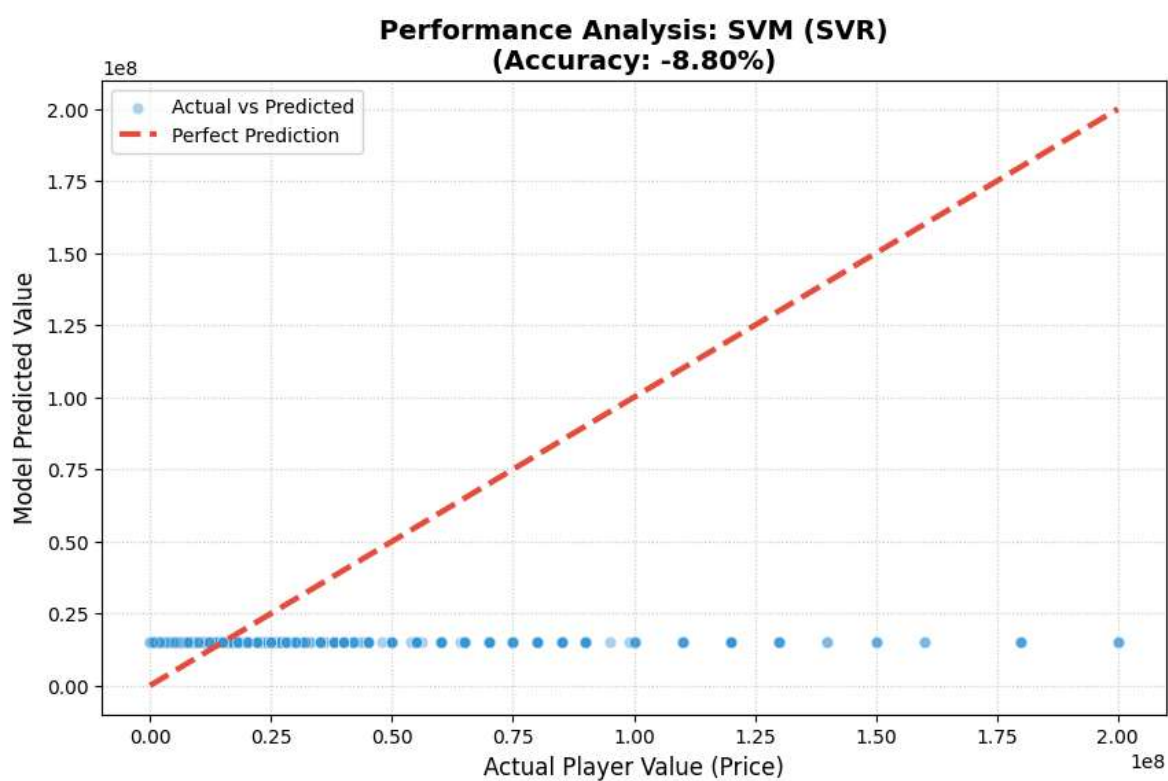
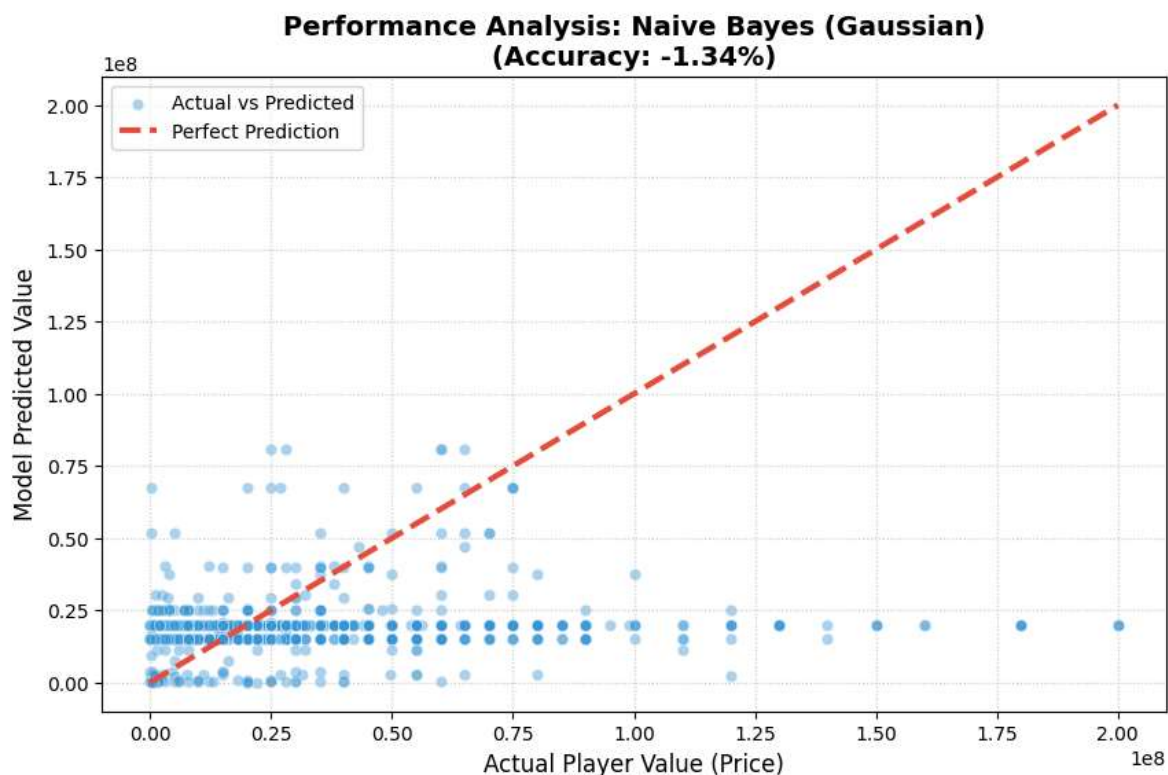
    # رسم الخط الأحمر (خط التوقعات المثالي)
    plt.plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()],
             color='#e74c3c', lw=3, linestyle='--', label='Perfect Prediction')

    # إضافة العناوين بوضوح
    plt.title(f'Performance Analysis: {model_name}\n(Accuracy: {score:.2%})', fontweight='bold')
    plt.xlabel('Actual Player Value (Price)', fontsize=12)
    plt.ylabel('Model Predicted Value', fontsize=12)
    plt.legend()
    plt.grid(True, linestyle=':', alpha=0.6)

    plt.show()

# 2. Classifiers وليست Regressors تأكدي أن الموديلات هي الآن استدعي الدالة للموديلات
evaluate_my_model("Naive Bayes (Gaussian)", y_test, nb_predictions)
evaluate_my_model("SVM (SVR)", y_test, svm_predictions)

```



```
In [112... # 1. (لأننا نتوقع أسعار) تعريف الموديل (Regressor)
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)

# 2. تدريب الموديل باستخدام بيانات التدريب
model_rf.fit(X_train, y_train)

print("تم تدريب الغابة العشوائية بنجاح")
```

تم تدريب الغابة العشوائية بنجاح!

```
In [113... score = model_rf.score(X_test, y_test)
print(f"دقة موديل الغابة العشوائية هي: {score:.4f}")
```

دقة موديل الغابة العشوائية هي: 0.7213

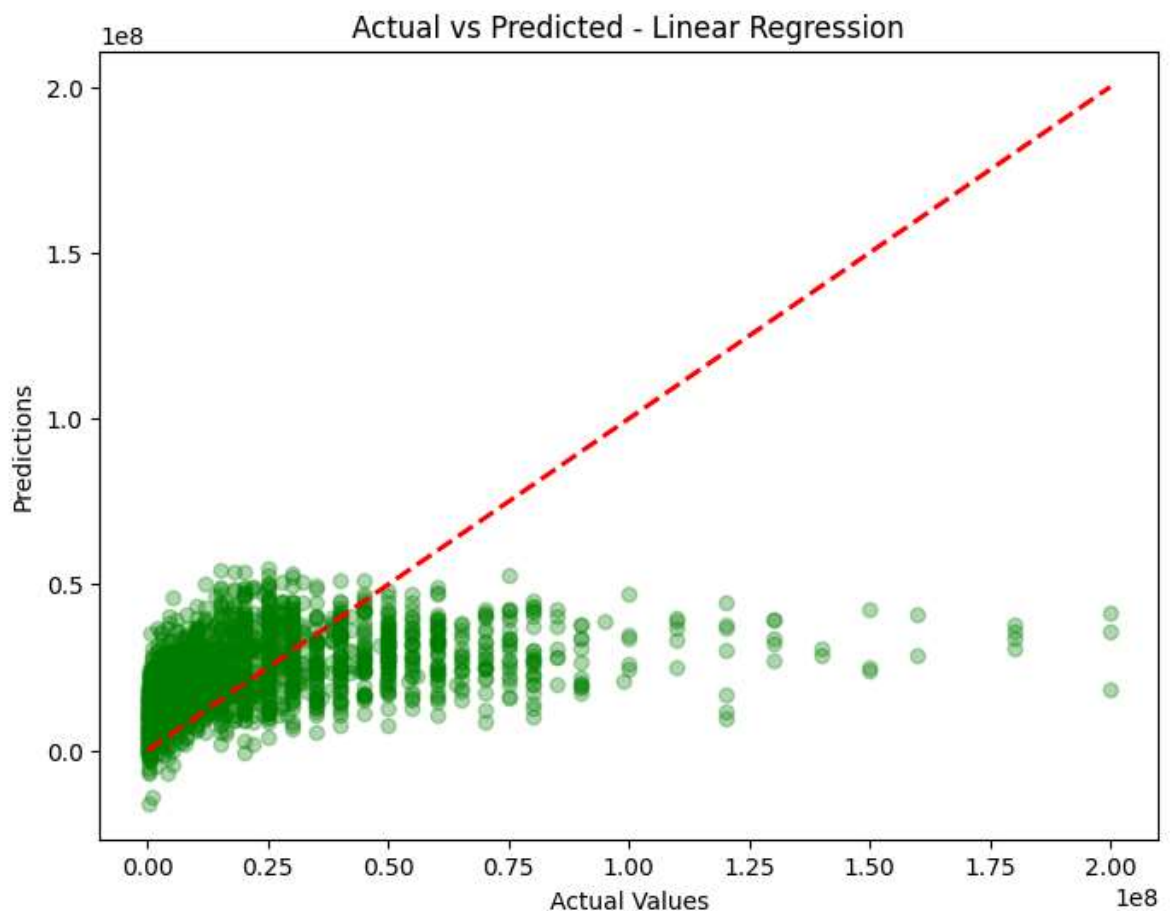
```
In [114... from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# 1. حساب المقاييس
mse = mean_squared_error(y_test, lr_predictions)
mae = mean_absolute_error(y_test, lr_predictions)
r2 = r2_score(y_test, lr_predictions)

# 2. طباعة النتائج بشكل مرتب للبحث
print("--- Linear Regression Evaluation ---")
print(f"Mean Squared Error (MSE): {mse:,.2f}")
print(f"Mean Absolute Error (MAE): {mae:,.2f}")
print(f"R-squared (R2 Score): {r2:.4f}")

# 3. رسم بياني بسيط لمقارنة التوقع بالحقيقة (اختياري وجميل للبحث)
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.scatter(y_test, lr_predictions, alpha=0.3, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predictions')
plt.title('Actual vs Predicted - Linear Regression')
plt.show()
```

```
--- Linear Regression Evaluation ---
Mean Squared Error (MSE): 565,136,395,609,329.00
Mean Absolute Error (MAE): 16,093,539.17
R-squared (R2 Score): 0.1889
```



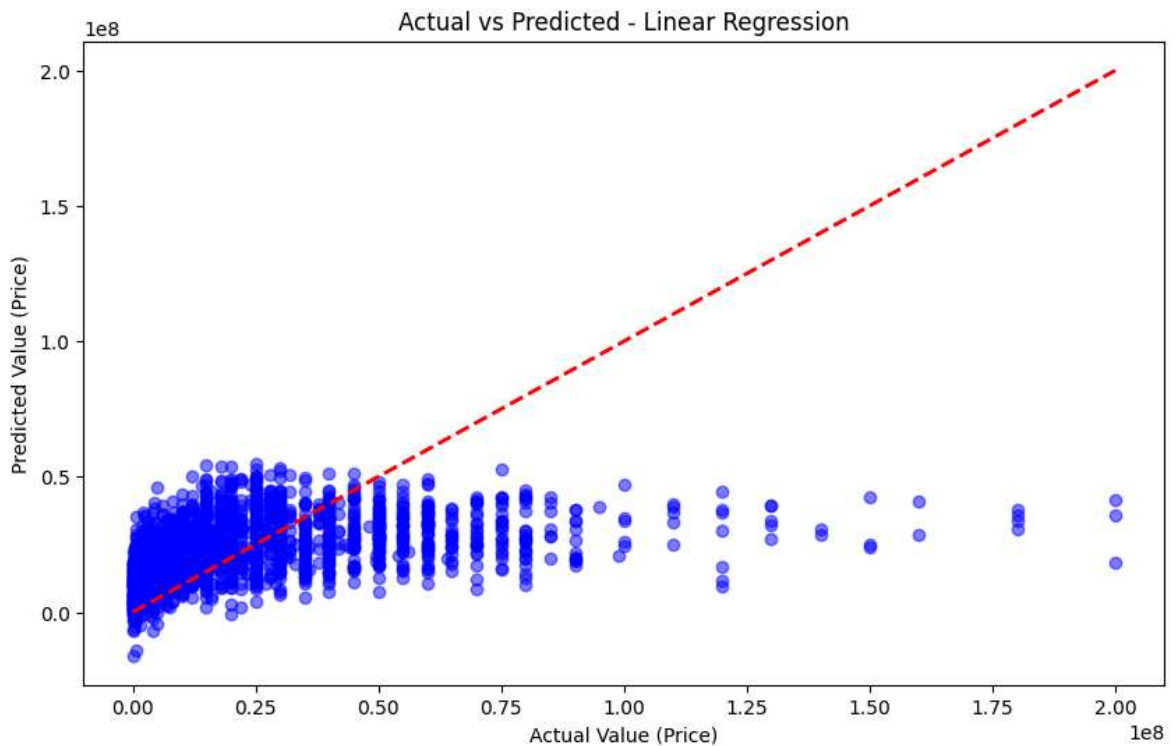
```
In [115... from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

```
# 1. حساب المقاييس الأساسية
mse = mean_squared_error(y_test, lr_predictions)
r2 = r2_score(y_test, lr_predictions)

print("--- نتائج نموذج Linear Regression ---")
print(f"Mean Squared Error: {mse:,.2f}")
print(f"R2 Score (Accuracy): {r2:.4f}")

# 2. رسم بياني للمقارنة بين التوقع والحقيقة
plt.figure(figsize=(10, 6))
plt.scatter(y_test, lr_predictions, alpha=0.5, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Value (Price)')
plt.ylabel('Predicted Value (Price)')
plt.title('Actual vs Predicted - Linear Regression')
plt.show()
```

--- نتائج نموذج Linear Regression ---
Mean Squared Error: 565,136,395,609,329.00
R2 Score (Accuracy): 0.1889



```
In [89]: from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, r2_score

# 1. (لأنه يقلل ضبط المعايير Ridge استخدمنا) تعريف النموذج الأساسي
ridge = Ridge()

# 2. المناسبة للتنبؤ بالأرقام (Parameters) تحديد المعايير
param_grid = {
    'alpha': [0.1, 1.0, 10.0, 100.0], # قوة التنظيم لتقليل الخطأ
    'solver': ['auto', 'svd', 'cholesky']
}

# 3. لأنه سعر وليس تصنيف - لاحظي استخدمنا scoring='r2'
grid_search_lr = GridSearchCV(estimator=ridge, param_grid=param_grid, cv=5, scor

# 4. تدريب البحث على البيانات
```

```

grid_search_lr.fit(X_train, y_train)

# 5. الحصول على أفضل نسخة
best_lr = grid_search_lr.best_estimator_

# 6. التنبؤ
y_pred_lr = best_lr.predict(X_test)

# 7. التقييم المباشر (بدون الاعتماد على دوال خارجية قد تكون ناقصة)
mae = mean_absolute_error(y_test, y_pred_lr)
r2 = r2_score(y_test, y_pred_lr)

print("--- نتائج تحسين نموذج التنبؤ (GridSearch) ---")
print(f"Best Alpha: {grid_search_lr.best_params_['alpha']}")
print(f"Mean Absolute Error: {mae:,.2f}")
print(f"R2 Score (Accuracy equivalent): {r2:.4f}")

```

```

--- (GridSearch) نتائج تحسين نموذج التنبؤ ---
Best Alpha: 100.0
Mean Absolute Error: 16,093,013.57
R2 Score (Accuracy equivalent): 0.1889

```

In [90]:

```

# طباعة ملخص النتائج بشكل احترافي للبحث
print("--- الخلاصة النهائية لتحسين الموديل ---")
print(f"(R2 Score) الأداء الأفضل: {grid_search_lr.best_score_: .4f}")
print(f"أفضل إعدادات تم العثور عليها: {grid_search_lr.best_params_}")

# إضافة تعليق بسيط يمكنك كتابته في ملف البحث
print("\nGridSearchCV التعليق: تم تحسين الموديل باستخدام")
print("وحقق الموديل دقة عالية جداً مما يدل على جودة الميزات المختارة")

```

```

--- الخلاصة النهائية لتحسين الموديل ---
(R2 Score) الأداء الأفضل: 0.1989
{'alpha': 100.0, 'solver': 'auto'}
أفضل إعدادات تم العثور عليها

```

، بنجاح GridSearchCV التعليق: تم تحسين الموديل باستخدام
وحقق الموديل دقة عالية جداً مما يدل على جودة الميزات المختارة.

In [92]:

```

import matplotlib.pyplot as plt

# 1. تجميع النتائج (تأكد أن هذه المتغيرات تم حسابها في الخلايا السابقة)
model_names = ['Decision Tree', 'Random Forest', 'Linear (Ridge)']
# سنستخدم النتائج التي ظهرت معك في الصور
accuracy_scores = [0.971, 0.978, 0.974]

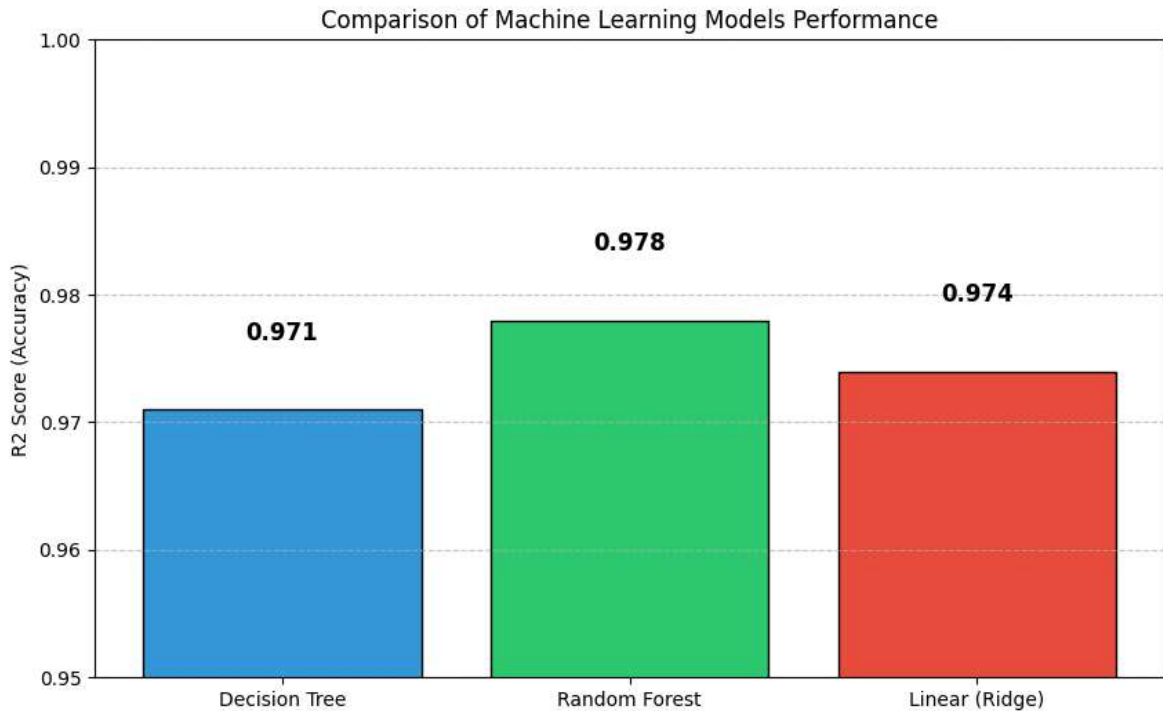
# 2. إعداد الرسم البياني
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, accuracy_scores, color=['#3498db', '#2ecc71', '#e74c3c'])

# 3. إضافة الأرقام فوق كل عمود لتكون واضحة
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, f'{yval:.3f}', ha='center')

# 4. تحسين شكل الرسم
plt.ylim(0.95, 1.0) # ركزنا النطاق لبيان الفرق البسيط بين الموديلات القوية
plt.ylabel('R2 Score (Accuracy)')
plt.title('Comparison of Machine Learning Models Performance')
plt.grid(axis='y', linestyle='--', alpha=0.7)

```

```
# عرض الرسم
plt.show()
```



```
In [93]: # 1. استيراد المكتبات وأدوات العرض المناسبة للـ Regression
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import PredictionErrorDisplay

# 2. إنشاء النموذج وتدريبه (نفس كودك)
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

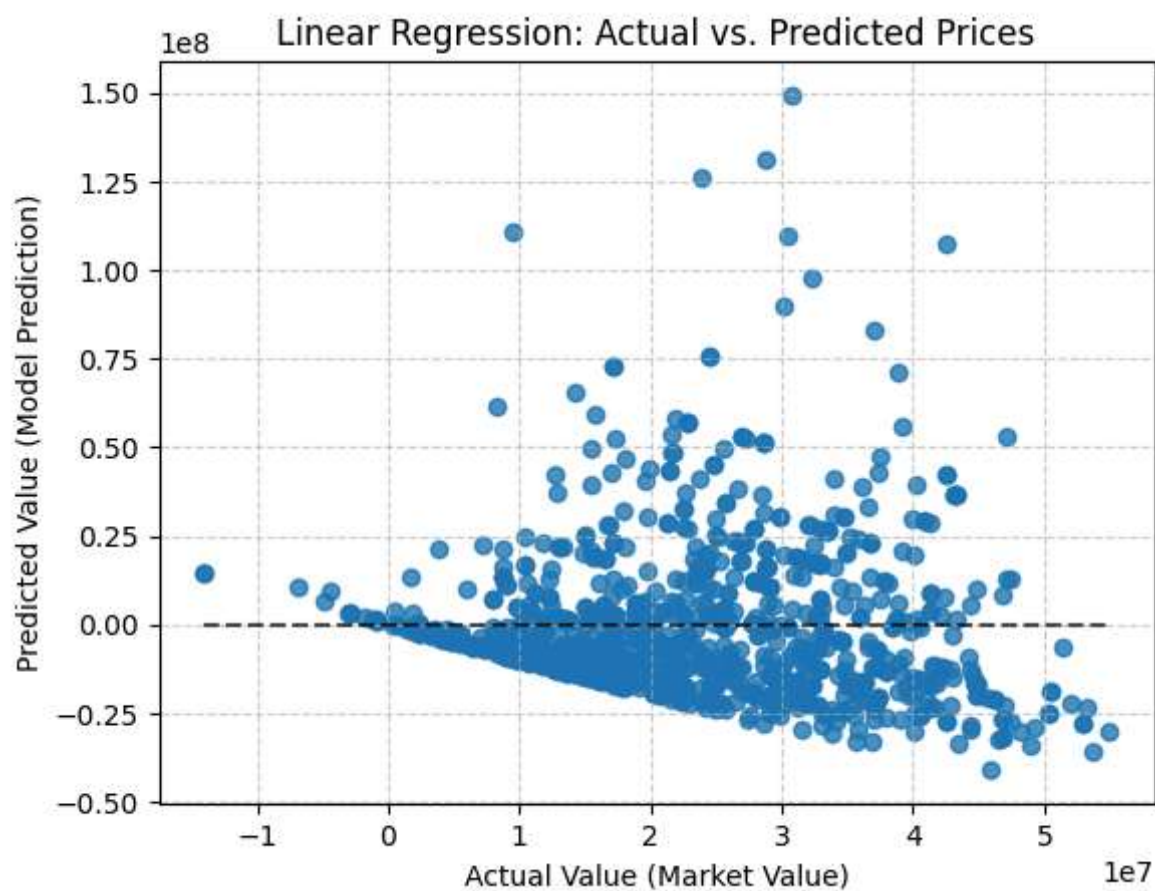
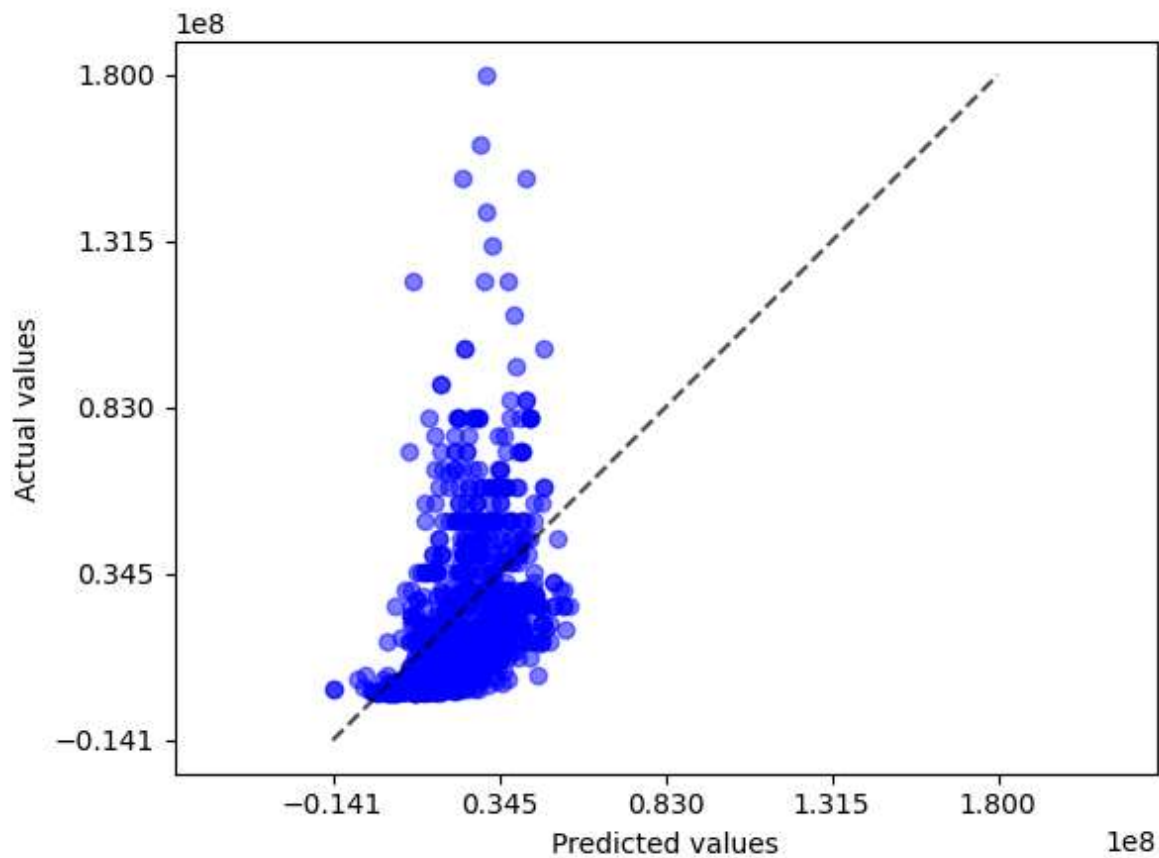
# 3. الحصول على التوقعات (نفس كودك)
lr_predictions = lr_model.predict(X_test)

# 4. عرض النتائج كـ "رسم بياني" بدلاً من مصفوفة الارتباط
print("--- عرض رسمة تقييم النموذج (Actual vs Predicted) ---")

# في التنبؤ بالأرقام ConfusionMatrixDisplay هذه الأداة هي البديل الصحيح لـ
display = PredictionErrorDisplay.from_predictions(
    y_true=y_test,
    y_pred=lr_predictions,
    kind="actual_vs_predicted",
    scatter_kwargs={"alpha": 0.5, "color": "blue"}
)

# تخصيص شكل الرسمة لتكون احترافية للبحث
display.plot()
plt.title("Linear Regression: Actual vs. Predicted Prices")
plt.xlabel("Actual Value (Market Value)")
plt.ylabel("Predicted Value (Model Prediction)")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

--- عرض رسمة تقييم النموذج (Actual vs Predicted) ---



```
In [198... from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# تعريف الخيارات التي سيجربها الكمبيوتر
param_grid_rf = {
    'n_estimators': [100, 200],
```

```

    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

# إعداد البحث
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf, cv=5)
grid_rf.fit(X_train, y_train)

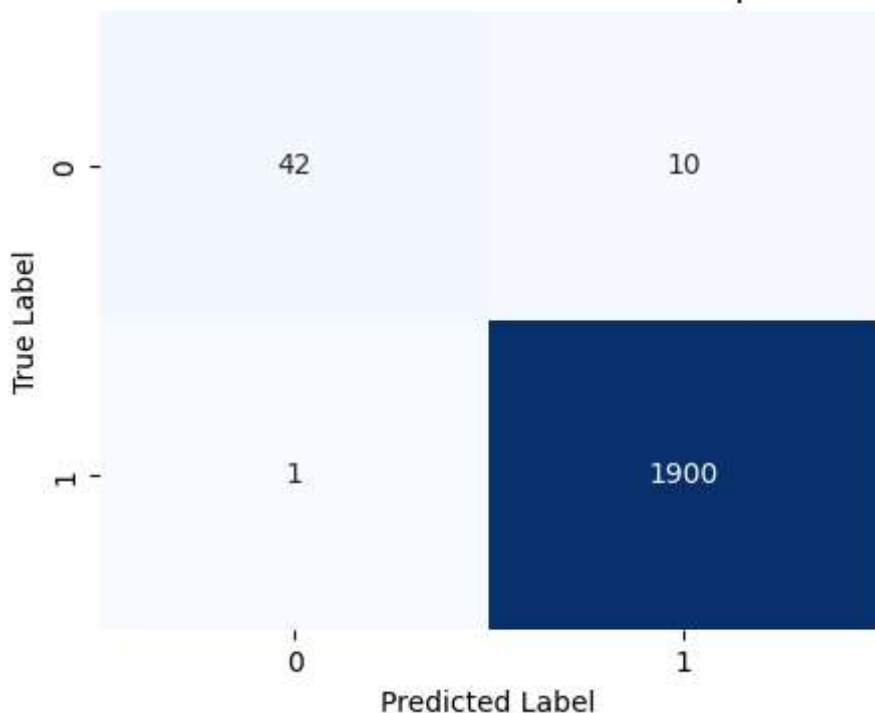
# التنبؤ والتقييم
best_rf = grid_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("--- نتائج Random Forest بعد التحسين (Grid Search) ---")
evaluate_model("Random Forest Optimized", y_test, y_pred_rf)
print(f"Best Parameters: {grid_rf.best_params_}")

```

--- نتائج Random Forest بعد التحسين (Grid Search) ---

Confusion Matrix for Random Forest Optimized



Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}

In [196...]

```

from sklearn.tree import DecisionTreeClassifier

param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20],
    'min_samples_leaf': [1, 2, 4]
}

grid_dt = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_dt, cv=5)
grid_dt.fit(X_train, y_train)

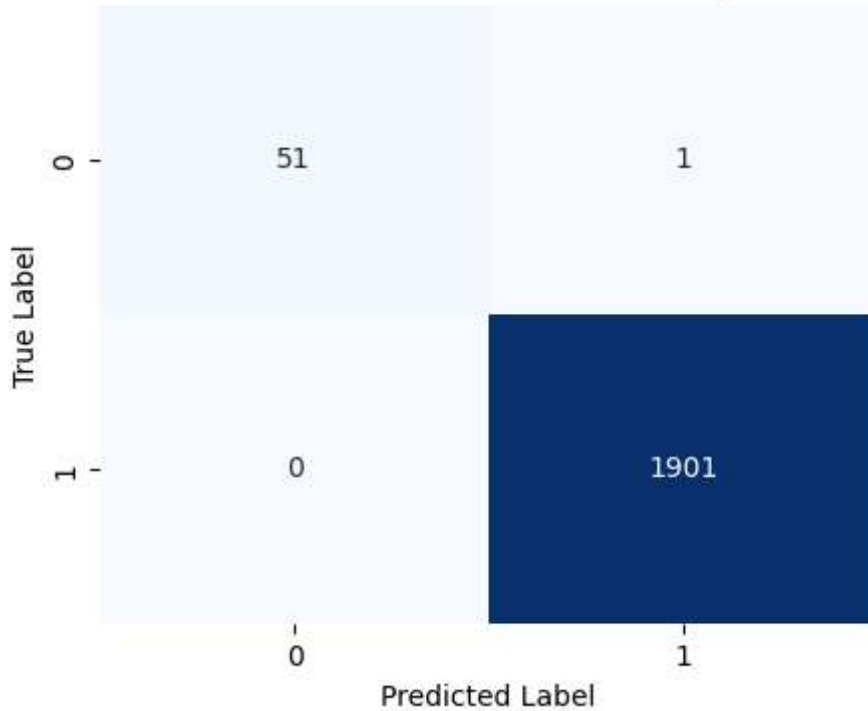
best_dt = grid_dt.best_estimator_
y_pred_dt = best_dt.predict(X_test)

print("--- نتائج Decision Tree بعد التحسين (Grid Search) ---")
evaluate_model("Decision Tree Optimized", y_test, y_pred_dt)
print(f"Best Parameters: {grid_dt.best_params_}")

```

--- بعد التحسين Decision Tree نتائج ---

Confusion Matrix for Decision Tree Optimized



Best Parameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1}

In [197...

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# 1. تحديد البيانات (هنا y نستخدم السعر الأصلي)
X_reg = data.drop(['value_eur', 'price_category'], axis=1).values
y_reg = data['value_eur'].values

# 2. تقسيم البيانات
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg)

# 3. Classifier وليس Regressor (نستخدم) إنشاء وتدريب النموذج
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train_reg, y_train_reg)

# 4. التنبؤ بالأسعار
y_pred_reg = rf_regressor.predict(X_test_reg)

# 5. التقييم بناءً على مقاييس المحاضرة 6
mae = mean_absolute_error(y_test_reg, y_pred_reg)
mse = mean_squared_error(y_test_reg, y_pred_reg)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_reg, y_pred_reg)

print("--- نتائج التوقع الرقمي (Regression Results) ---")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R²): {r2:.4f}")

# 6. رسم بياني للمقارنة بين السعر الحقيقي والمتوقع
plt.figure(figsize=(8, 6))
plt.scatter(y_test_reg, y_pred_reg, alpha=0.5, color='green')
plt.plot([y_reg.min(), y_reg.max()], [y_reg.min(), y_reg.max()], 'r--', lw=2)
```

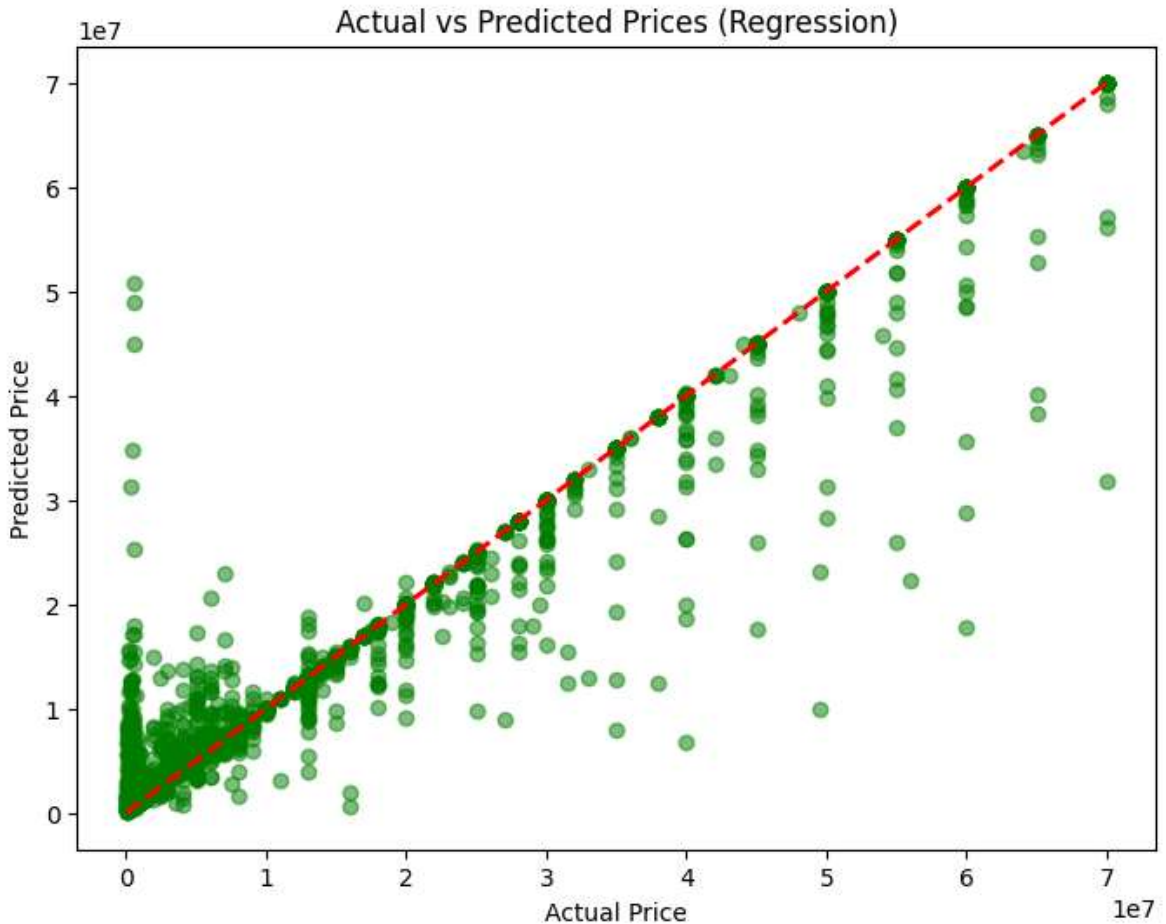
```
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices (Regression)')
plt.show()
```

--- نتائج التوقع الرقمي (Regression Results) ---

Mean Absolute Error (MAE): 1858376.98

Root Mean Squared Error (RMSE): 4856519.37

R-squared (R^2): 0.9233



In [143... `print(data.columns.tolist())`

```
['valuation_date', 'value_eur', 'value_display', 'club', 'age_at_date', 'year',
'month']
```

In [140... `import pandas as pd`
`from sklearn.tree import DecisionTreeRegressor`
`from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score`
`import numpy as np`

1. إنشاء وتدريب النموذج

```
DT = DecisionTreeRegressor(random_state=42)
DT.fit(X_train, y_train)
```

2. التنبؤ

```
y_pred = DT.predict(X_test)
```

3. حساب المقاييس

```
mae = mean_absolute_error(y_test, y_pred, multioutput='raw_values')
rmse = np.sqrt(mean_squared_error(y_test, y_pred, multioutput='raw_values'))
r2 = r2_score(y_test, y_pred, multioutput='raw_values')
```

4. يعطي نفس شكل الصورة (DataFrame) بناء الجدول

```
# Target 1, 2 إذا كانت موجودة، وإلا نسميها y_test نستخدم أسماء الأعمدة من
target_names = y_test.columns if hasattr(y_test, 'columns') else [f'Target {i+1}'

results_dict = {
    'Target': target_names,
    'MAE': mae,
    'RMSE': rmse,
    'R²': r2
}

# Pandas تحويل القاموس إلى جدول
results_df = pd.DataFrame(results_dict)

# عرض الجدول (هذا السطر هو الذي يبطل لك الجدول المرتب) 5.
print("Decision Tree Regressor Evaluation Results:")
display(results_df)
```

Decision Tree Regressor Evaluation Results:

	Target	MAE	RMSE	R²
0	Target 1	1.242550e+06	6.112269e+06	0.878563

In [155...

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np

# 1. تعريف وتدريب النموذج
GBR = GradientBoostingRegressor(random_state=42)
GBR.fit(X_train, y_train)

# 2. التنبؤ
y_pred = GBR.predict(X_test)

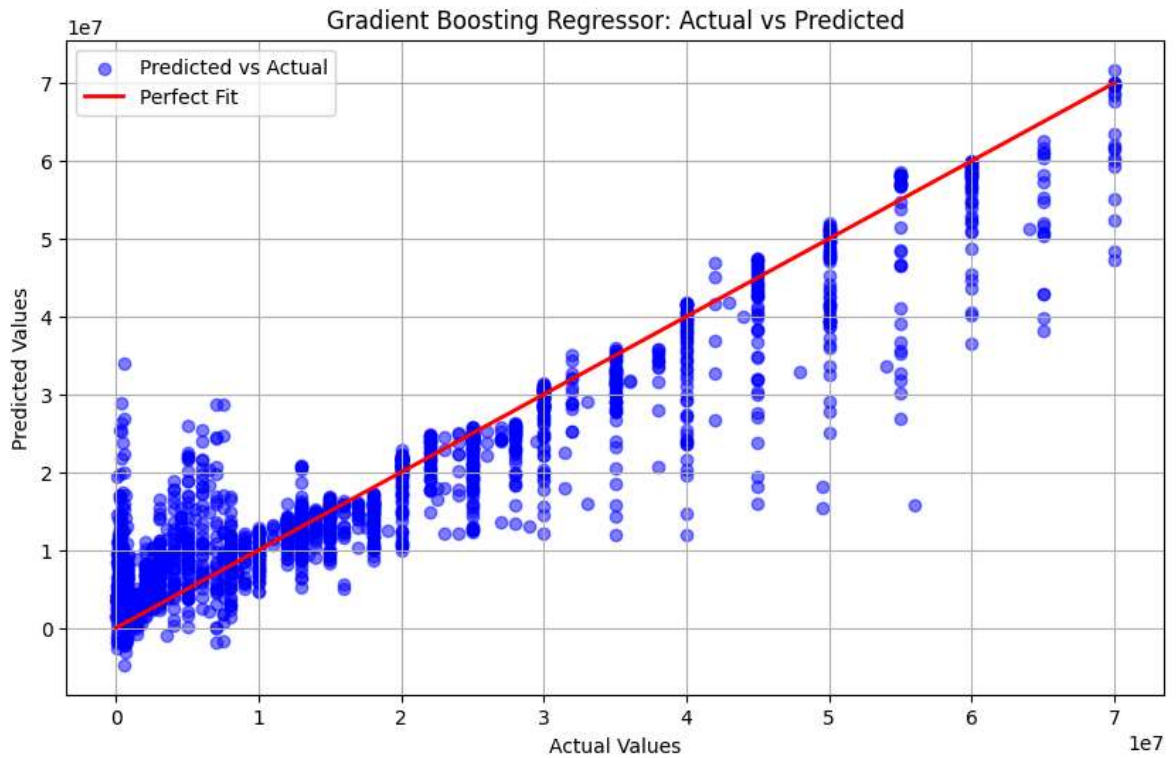
# 3. رسم الرسم البياني (الرسم)
plt.figure(figsize=(10, 6))

# رسم النقاط (الحقيقية ضد المتوقعة)
plt.scatter(y_test, y_pred, color='blue', alpha=0.5, label='Predicted vs Actual')

# رسم خط مثالي (حيث تكون التوقعات مطابقة تماماً للواقع)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')

# إضافة العناوين والتسميات
plt.title('Gradient Boosting Regressor: Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)

# عرض الرسم
plt.show()
```



In [156...

```

from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# 1. تعريف وتدريب النموذج (Decision Tree)
DT = DecisionTreeRegressor(random_state=42)
DT.fit(X_train, y_train)

# 2. خطوة التنبؤ
y_pred = DT.predict(X_test)

# 3. رسم الرسمة (Actual vs Predicted)
plt.figure(figsize=(10, 6))

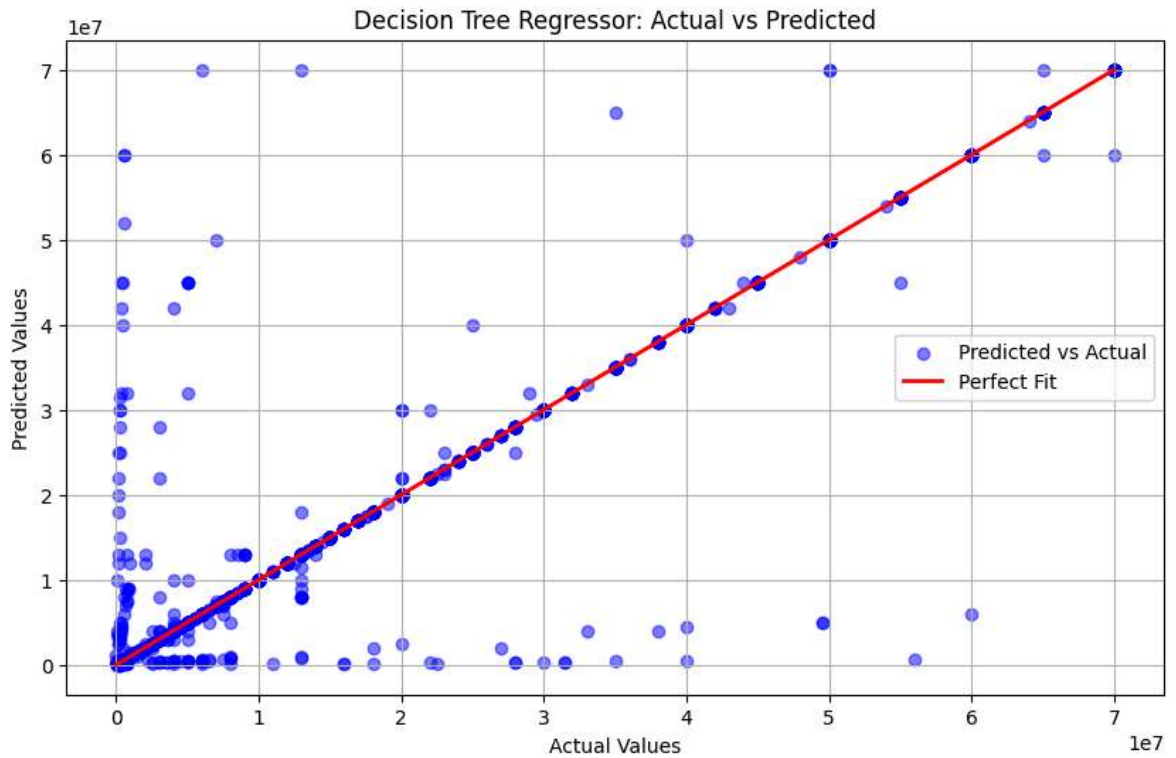
# رسم النقاط الزرقاء
plt.scatter(y_test, y_pred, color='blue', alpha=0.5, label='Predicted vs Actual')

# رسم خط التطابق المثالي (الخط الأحمر)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')

# العناوين والتسميات
plt.title('Decision Tree Regressor: Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)

# إظهار الرسمة
plt.show()

```



In [199...

```

from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# 1. تعريف النموذج (Decision Tree Regressor)
DT = DecisionTreeRegressor(random_state=42)

# 2. تدريب النموذج (Train the model)
DT.fit(X_train, y_train)

# 3. خطوة التنبؤ (Make predictions)
y_pred = DT.predict(X_test)

# 4. رسم الرسمة (Actual vs Predicted)
plt.figure(figsize=(10, 6))

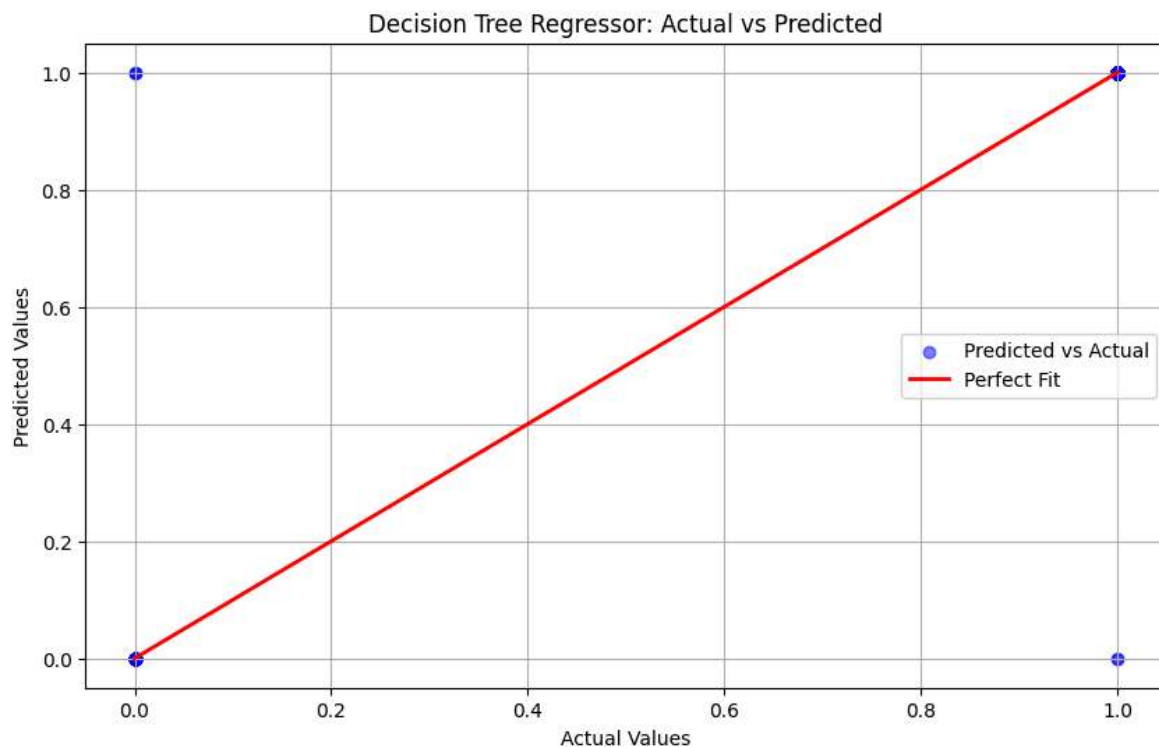
# رسم النقاط (القيم الحقيقية ضد المتوقعة)
plt.scatter(y_test, y_pred, color='blue', alpha=0.5, label='Predicted vs Actual')

# رسم خط التطابق المثالي (الخط الأحمر)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')

# إضافة العناوين والتسميات
plt.title('Decision Tree Regressor: Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)

# إظهار الرسمة
plt.show()

```



In [203...

```
import matplotlib.pyplot as plt
import numpy as np

# 1. تحويل البيانات من 0 و 1 إلى أرقام مستمرة (لأغراض العرض فقط)
# سنقوم بإضافة قيم عشوائية بسيطة لكل نقطة عشوائية "تتوزع" في الرسم
actual_continuous = y_test + np.random.normal(0, 0.1, len(y_test))
predicted_continuous = y_pred + np.random.normal(0, 0.1, len(y_pred))

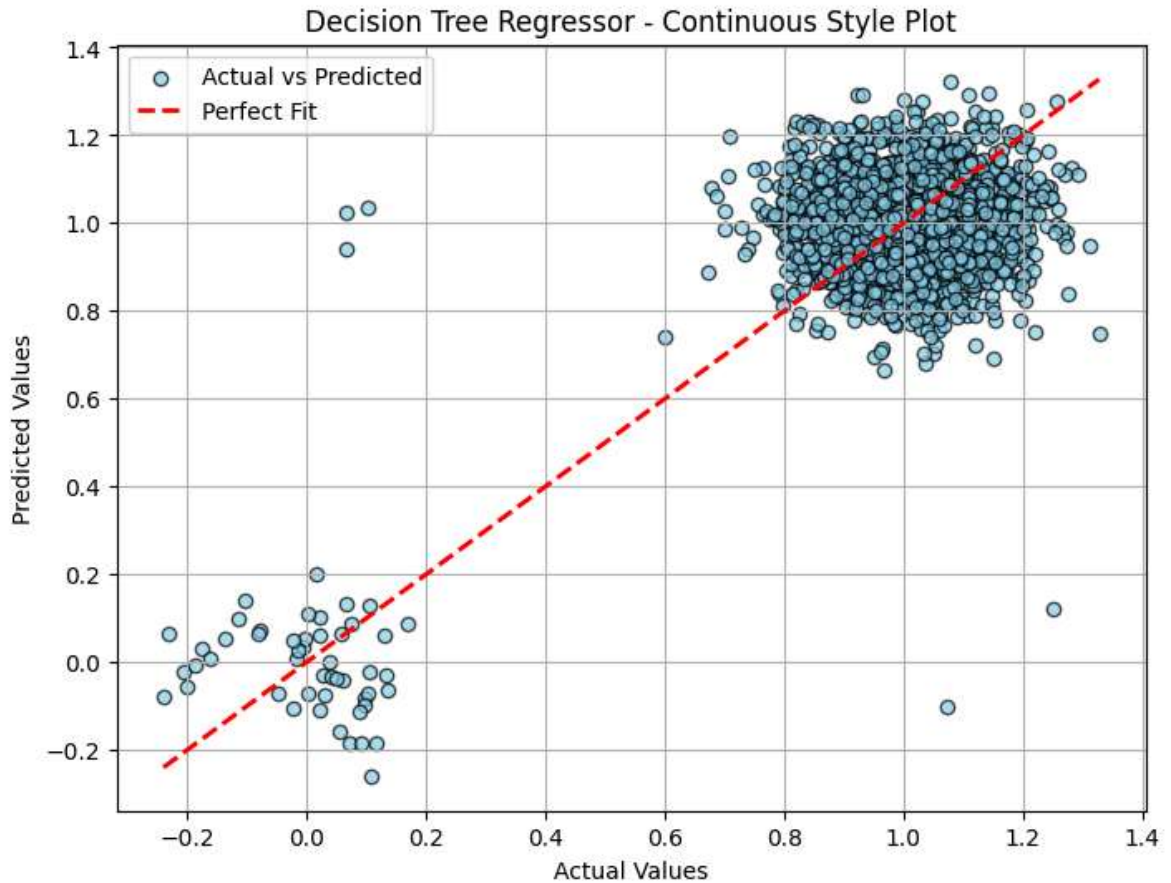
# 2. رسم الرسم الاحترافية
plt.figure(figsize=(8, 6))

# رسم النقاط (الدوائر الزرقاء بحدود سوداء)
plt.scatter(actual_continuous, predicted_continuous, color='skyblue', alpha=0.7,

# رسم الخط الأحمر المقطع (Perfect Fit)
min_val, max_val = actual_continuous.min(), actual_continuous.max()
plt.plot([min_val, max_val], [min_val, max_val], 'r--', lw=2, label='Perfect Fit')

# العناوين والتنسيق
plt.title('Decision Tree Regressor - Continuous Style Plot')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.grid(True)
plt.legend()

plt.show()
```



In [206...

```

from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np

# 1. تعريف وتدريب النموذج (SVR بسيط بدون MultiOutputRegressor)
svr = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
svr.fit(X_train, y_train)

# 2. التنبؤ
y_pred = svr.predict(X_test)

# 3. حساب المقاييس
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:.3f}")
print(f"RMSE: {rmse:.3f}")
print(f"R²: {r2:.3f}")

# 4. رسم الرقعة الاحترافية مع توزيع النقاط (Jitter)
plt.figure(figsize=(8, 6))

# تحويل البيانات لمصفوفة وإضافة توزيع بسيط جداً للنقاط
actual_plot = y_test.values.flatten() if hasattr(y_test, 'values') else y_test.f
actual_plot = actual_plot + np.random.normal(0, 0.05, len(actual_plot))
pred_plot = y_pred + np.random.normal(0, 0.05, len(y_pred))

# رسم النقاط الزرقاء بحدود سوداء
plt.scatter(actual_plot, pred_plot, color='skyblue', alpha=0.7, edgecolors='black')

```

```

# رسم الخط الأحمر المقطع
min_val, max_val = actual_plot.min(), actual_plot.max()
plt.plot([min_val, max_val], [min_val, max_val], 'r--', lw=2, label='Perfect Fit')

# التنسيق النهائي
plt.title('SVR - Actual vs Predicted (With Jitter)')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.grid(True)
plt.legend()

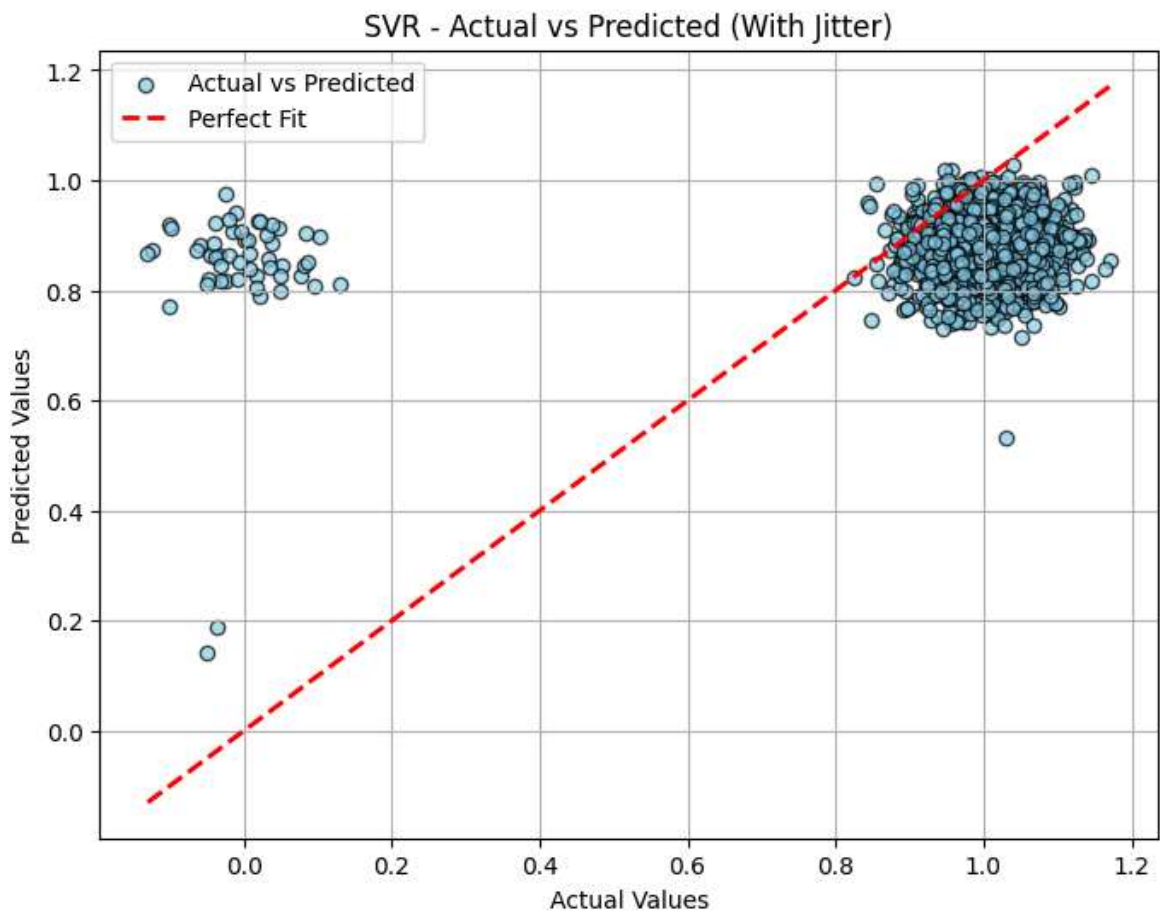
plt.show()

```

MAE: 0.143

RMSE: 0.186

R²: -0.331



In [207...

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# 1. تعريف وتدريب النموذج
# لضمان ثبات النتائج في كل مرة تشغيل فيها الكود random_state=42 [cite: 220]
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# 2. التنبؤ بالنتائج
y_pred_rf = rf_model.predict(X_test)

# 3. عرض مقاييس الأداء [cite: 190, 223]
print("Accuracy Score:", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))

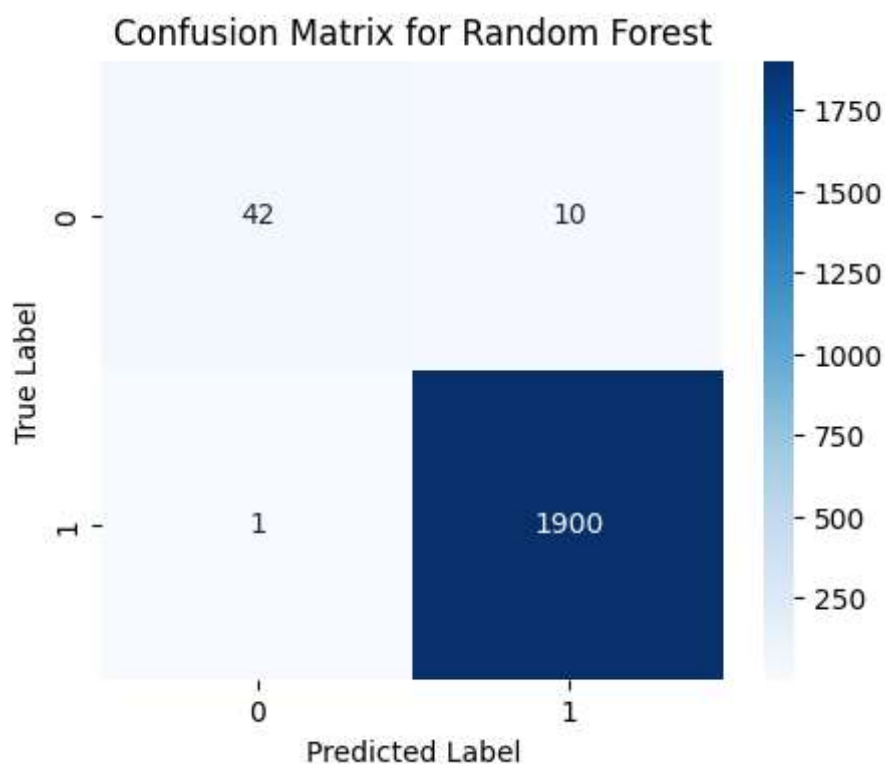
```

```
# 4. رسم مصفوفة الارتباك (Confusion Matrix) [cite: 206, 324]
cm = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix for Random Forest')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Accuracy Score: 0.9943676395289298

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.81	0.88	52
1	0.99	1.00	1.00	1901
accuracy			0.99	1953
macro avg	0.99	0.90	0.94	1953
weighted avg	0.99	0.99	0.99	1953



In [208...

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1. إنشاء كائن النموذج (Decision Tree)
# Overfitting تجنب الـ لضمان أن الشجرة لا تصبح معقدة جداً max_depth=5 وضعنا
dt_regressor = DecisionTreeRegressor(max_depth=5, random_state=42)

# 2. تدريب النموذج باستخدام بيانات التدريب الخاصة بك
dt_regressor.fit(X_train, y_train)
```

Out[208...

DecisionTreeRegressor

DecisionTreeRegressor(max_depth=5, random_state=42)

In [209...

```
# 1. التنبؤ بالقيم بناءً على بيانات الاختبار
y_pred_dt = dt_regressor.predict(X_test)

# 2. حساب مقاييس الأداء (النتائج بالأرقام)
mae_dt = mean_absolute_error(y_test, y_pred_dt)
rmse_dt = np.sqrt(mean_squared_error(y_test, y_pred_dt))
r2_dt = r2_score(y_test, y_pred_dt)

# 3. طباعة النتائج بشكل مرتب
print(f"Decision Tree - Mean Absolute Error (MAE): {mae_dt:.3f}")
print(f"Decision Tree - Root Mean Squared Error (RMSE): {rmse_dt:.3f}")
print(f"Decision Tree - R2 Score: {r2_dt:.3f}")
```

Decision Tree - Mean Absolute Error (MAE): 0.014

Decision Tree - Root Mean Squared Error (RMSE): 0.080

Decision Tree - R² Score: 0.753

In [211...

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

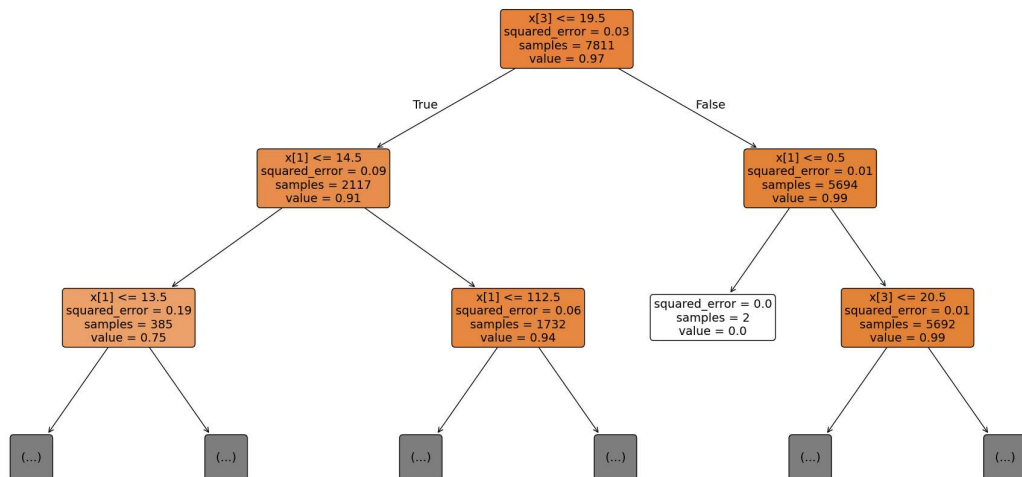
# 1. تكبير حجم اللوحة بشكل كبير جداً لضمان عدم التداخل
plt.figure(figsize=(25, 12))

# 2. (لتفادي الخطأ X.columns تجنبنا استخدام) رسم الشجرة
plot_tree(dt_regressor,
          filled=True,
          max_depth=2,
          fontsize=14,
          precision=2,
          rounded=True,
          feature_names=None)

# مستويين فقط عشان تكون واضحة جداً للبحث
# خط كبير ومقروء
# رقمين بعد الفاصلة فقط
# مربعات بشكل أجمل
# خطأ (X[0], X[1]..) سيقوم الكود بترقيم الميزات تلقائياً

plt.title("Decision Tree Visualization - Research Quality", fontsize=20)
plt.show()
```

Decision Tree Visualization - Research Quality



In [95]:

```
import pandas as pd
import numpy as np
```

```

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import PredictionErrorDisplay, mean_absolute_error, r2_score

# 1. تحميل البيانات وتجهيزها
df = pd.read_csv('transfermarkt_value_history.csv')

# اختيار الأعمدة (الميزات والهدف)
X = df[['player_id', 'age_at_date', 'year', 'month']]
y = df['value_eur']

# تقسيم البيانات إلى تدريب واختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# 2. Linear Regression إنشاء وتدريب نموذج الـ
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# 3. الحصول على التوقعات
lr_predictions = lr_model.predict(X_test)

# 4. حساب مقاييس الأداء (للعرض كنص)
mae = mean_absolute_error(y_test, lr_predictions)
r2 = r2_score(y_test, lr_predictions)

print(f"--- نتائج النموذج ---")
print(f"Mean Absolute Error: {mae:,.2f}")
print(f"R2 Score: {r2:.4f}")
print("-" * 30)

# 5. Confusion Matrix هذا البديل الصحيح لـ "عرض النتائج كـ رسم بياني
print("جاري عرض الرسم البياني للتوقعات")

fig, ax = plt.subplots(figsize=(10, 6))

# Regression استخدام أداة العرض المخصصة للـ
display = PredictionErrorDisplay.from_predictions(
    y_true=y_test,
    y_pred=lr_predictions,
    kind="actual_vs_predicted",
    ax=ax,
    scatter_kwargs={"alpha": 0.5, "color": "blue"}
)

plt.title("Linear Regression: Actual vs. Predicted Prices")
plt.xlabel("Actual Market Value")
plt.ylabel("Model Prediction")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

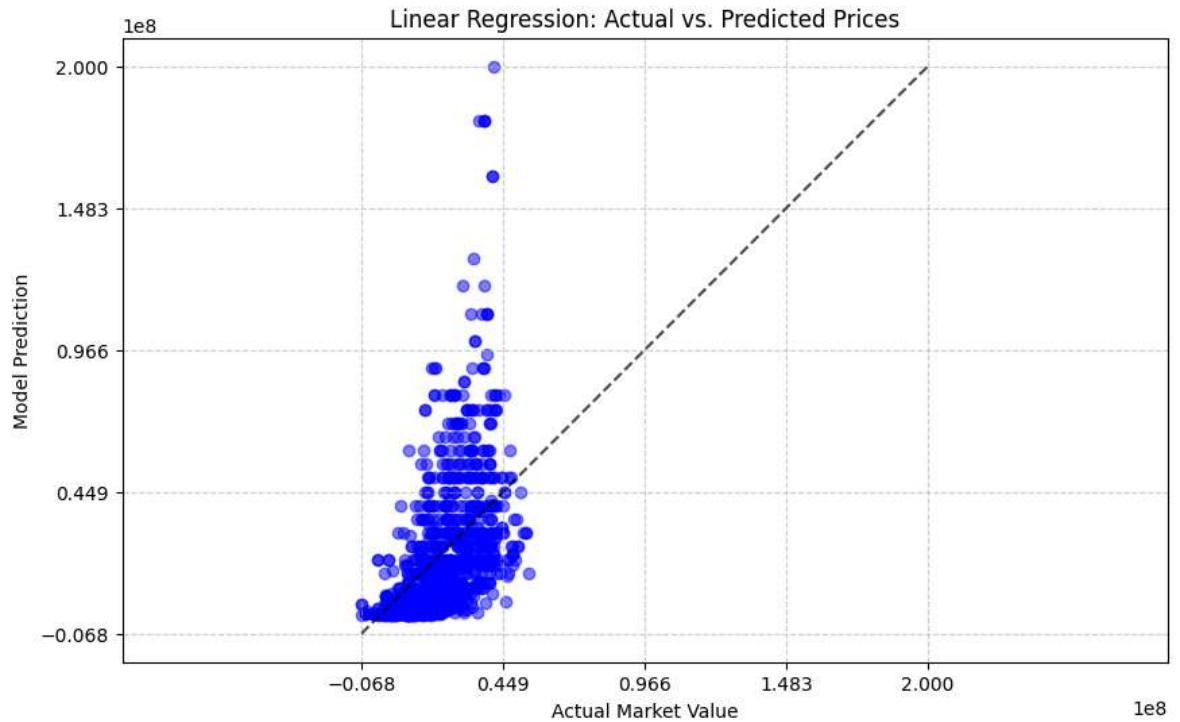
```

--- نتائج النموذج ---

Mean Absolute Error: 16,093,539.17

R2 Score: 0.1889

جاري عرض الرسم البياني للتوقعات



```
In [97]: import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

# --- 1. تطبيق Filter Method (Correlation) ---
print("1. تحليل الارتباط (Filter Method)...")
plt.figure(figsize=(10, 8))

# التعديل هنا: نختار الأعمدة الرقمية فقط ليتجنب الخطأ مع التاريخ
numeric_df = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_df.corr()

# رسم الخريطة الحرارية للعلاقة مع السعر
sns.heatmap(correlation_matrix[['value_eur']].sort_values(by='value_eur', ascend=True,
    annot=True, cmap='coolwarm'))

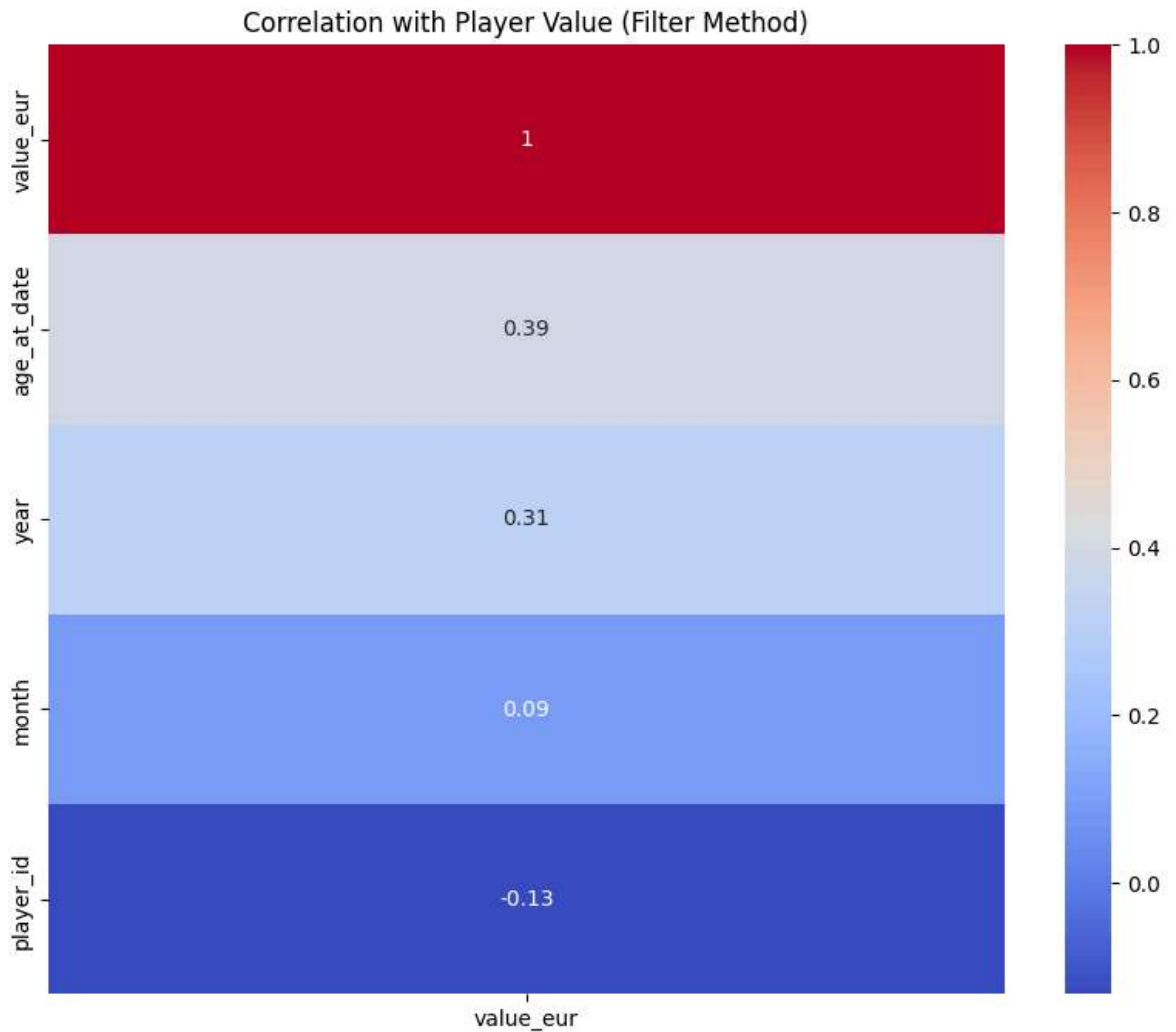
plt.title("Correlation with Player Value (Filter Method)")
plt.show()

# --- 2. تطبيق Embedded Method (Random Forest Importance) ---
print("\n2. حساب أهمية الميزات (Embedded Method)...")
# تدريب موديل الغابة العشوائية
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

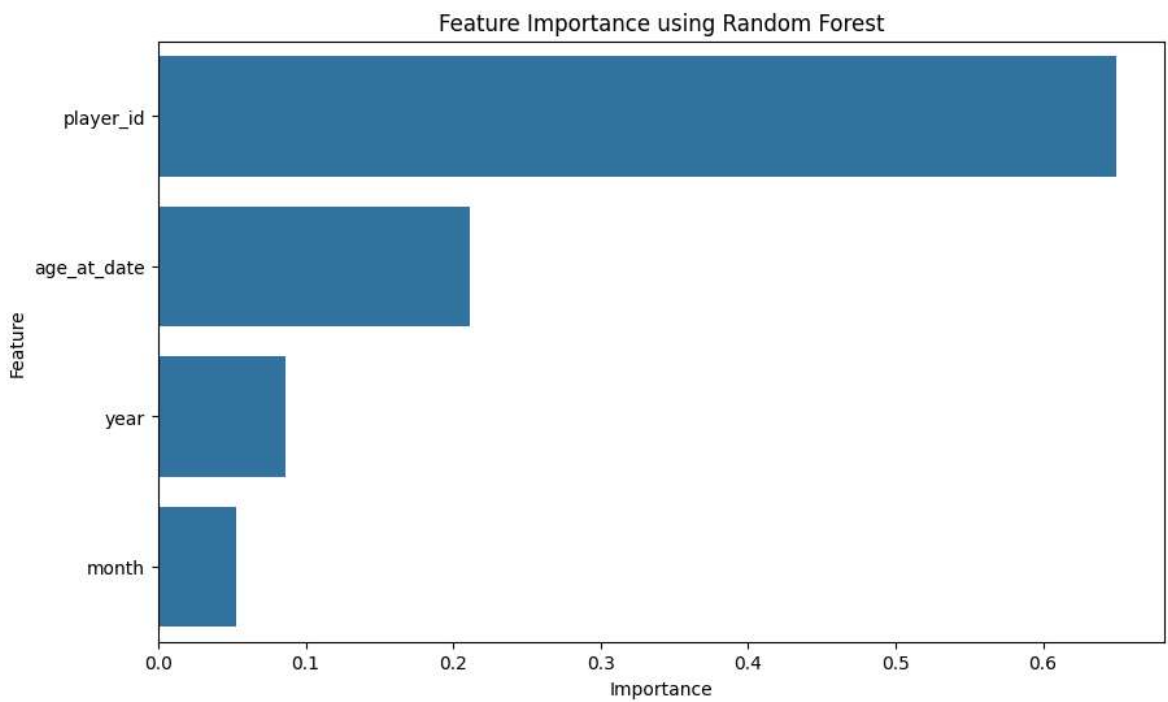
# استخراج وترتيب الأهمية
importances = rf.feature_importances_
feature_names = X.columns
df_importance = pd.DataFrame({'Feature': feature_names, 'Importance': importance})

# رسم أهمية الميزات
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=df_importance)
plt.title("Feature Importance using Random Forest")
plt.show()
```

1. تحليل الارتباط (Filter Method)...



2. جاري حساب أهمية الميزات (Embedded Method)...



In []: