OLS_RF_secondary

October 22, 2024

1 Secondary Dataset

2 Linear Regression and Price Analysis using Random Forest

[37]:
```
# Load necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
from matplotlib import gridspec
import math
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

sns.set()
```

[38]:
```
# set this to either primary or secondary depending on the analysis you want to
 ↪do
dataset = 'secondary'

data = pd.read_csv(f'../dataset/mapped_{dataset}.csv', sep=",")
transform = pd.read_csv(f'../dataset/transform_{dataset}.csv', sep=",")
data = data.drop(columns=['uuid'])
```

[39]:
```
import statsmodels.api as sm

X = data.loc[:, ~data.columns.isin(['price'])]
y = data["price"]

model = sm.OLS(y, X)     # Describe model

result = model.fit()       # Fit model

print(result.summary())
```

```
                            OLS Regression Results
================================================================================
```

```
=======
Dep. Variable:                  price   R-squared (uncentered):
0.491
Model:                            OLS   Adj. R-squared (uncentered):
0.491
Method:                 Least Squares   F-statistic:
9177.
Date:                Tue, 22 Oct 2024   Prob (F-statistic):
0.00
Time:                        03:18:39   Log-Likelihood:
-2.9825e+05
No. Observations:              228637   AIC:
5.966e+05
Df Residuals:                  228613   BIC:
5.968e+05
Df Model:                          24
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
artist           -1.072e-05   4.62e-06     -2.320      0.020   -1.98e-05
-1.66e-06
cardFinish          -0.3282      0.004    -85.104      0.000      -0.336
-0.321
colorIdentity       -0.0017      0.001     -1.897      0.058      -0.004
5.76e-05
colors            -5.21e-06      0.001     -0.006      0.995      -0.002
0.002
edhrecRank       -3.306e-05   2.58e-07   -128.141      0.000   -3.36e-05
-3.26e-05
edhrecSaltiness      0.7293      0.007     98.973      0.000       0.715
0.744
gameAvailability     0.8596      0.007    118.211      0.000       0.845
0.874
isReprint           -0.0200      0.004     -4.985      0.000      -0.028
-0.012
language             0.2492      0.007     35.434      0.000       0.235
0.263
layout               0.0510      0.002     25.889      0.000       0.047
0.055
manaCost             0.0002   1.12e-05     15.915      0.000       0.000
0.000
manaValue            0.0143      0.001     12.591      0.000       0.012
0.017
name              1.449e-06   3.67e-07      3.945      0.000    7.29e-07
```

```
2.17e-06
number            7.003e-06    1.5e-06      4.684      0.000    4.07e-06
9.93e-06
originalType      4.436e-05   3.63e-06     12.221      0.000    3.72e-05
5.15e-05
power                0.0036      0.001      5.943      0.000       0.002
0.005
priceProvider       -0.0993      0.003    -37.689      0.000      -0.104
-0.094
providerListing     -0.2784      0.006    -44.710      0.000      -0.291
-0.266
rarity              -0.0054      0.001     -4.298      0.000      -0.008
-0.003
setCode              0.0003   1.22e-05     21.954      0.000       0.000
0.000
supertypes          -0.0682      0.004    -18.484      0.000      -0.075
-0.061
toughness            0.0024      0.001      3.762      0.000       0.001
0.004
type             -6.447e-05   6.41e-06    -10.064      0.000    -7.7e-05
-5.19e-05
types               -0.0040      0.001     -4.165      0.000      -0.006
-0.002
==============================================================================
Omnibus:                    83948.314   Durbin-Watson:                   1.219
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           283564.391
Skew:                           1.898   Prob(JB):                         0.00
Kurtosis:                       6.918   Cond. No.                     7.03e+04
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 7.03e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[40]:
```python
Xt = transform.loc[:, ~transform.columns.isin(['price'])]
yt = transform["price"]

modelt = sm.OLS(yt, Xt)    # Describe model

resultt = modelt.fit()       # Fit model

print(resultt.summary())
```

```
                            OLS Regression Results
```

```
================================================================================
=======
Dep. Variable:                  price   R-squared (uncentered):
0.962
Model:                            OLS   Adj. R-squared (uncentered):
0.962
Method:                 Least Squares   F-statistic:
2.491e+05
Date:                Tue, 22 Oct 2024   Prob (F-statistic):
0.00
Time:                        03:18:41   Log-Likelihood:
4.6828e+05
No. Observations:              228637   AIC:
-9.365e+05
Df Residuals:                  228614   BIC:
-9.363e+05
Df Model:                          23
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
artist            1.397e-06   1.62e-07      8.644      0.000    1.08e-06
1.71e-06
cardFinish           0.0029      0.000     21.579      0.000       0.003
0.003
colors            3.052e-05   6.22e-06      4.905      0.000    1.83e-05
4.27e-05
edhrecRank       -3.513e-07   9.03e-09    -38.921      0.000   -3.69e-07
-3.34e-07
edhrecSaltiness      0.2070      0.000    802.646      0.000       0.206
0.207
gameAvailability     0.0025      0.000      9.852      0.000       0.002
0.003
isReprint            0.0004      0.000      2.965      0.003       0.000
0.001
language             0.0143      0.000     58.228      0.000       0.014
0.015
layout               0.0043   6.89e-05     61.858      0.000       0.004
0.004
manaCost          8.304e-06   3.91e-07     21.236      0.000    7.54e-06
9.07e-06
manaValue            0.0013   3.98e-05     33.731      0.000       0.001
0.001
name              7.602e-08   1.29e-08      5.915      0.000    5.08e-08
1.01e-07
```

```
number              5.418e-07    5.23e-08     10.356      0.000     4.39e-07
6.44e-07
originalType       -1.266e-06    1.27e-07     -9.963      0.000    -1.51e-06
-1.02e-06
power               3.774e-05    2.14e-05      1.763      0.078    -4.21e-06
7.97e-05
priceProvider          0.0003    9.22e-05      2.897      0.004     8.64e-05
0.000
providerListing        0.0025       0.000     11.690      0.000        0.002
0.003
rarity                 0.0010    4.36e-05     23.427      0.000        0.001
0.001
setCode             2.861e-06    4.26e-07      6.721      0.000     2.03e-06
3.7e-06
supertypes             0.0022       0.000     16.740      0.000        0.002
0.002
toughness           9.319e-05    2.21e-05      4.208      0.000     4.98e-05
0.000
type                7.093e-06    2.24e-07     31.646      0.000     6.65e-06
7.53e-06
types                 -0.0007    3.32e-05    -19.836      0.000       -0.001
-0.001
==============================================================================
Omnibus:                   120507.106   Durbin-Watson:                   0.517
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2104047.035
Skew:                          -2.144   Prob(JB):                         0.00
Kurtosis:                      17.229   Cond. No.                     7.03e+04
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 7.03e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[41]:
```
from sklearn.metrics import mean_squared_error

y_pred = result.predict(X)
mse = mean_squared_error(y, y_pred)

fstat, pvalue = sm.stats.linear_rainbow(result)
print(f' f stat: {fstat} | p value: {pvalue} | mse: {mse}')
```

```
 f stat: 1.1950485339398464 | p value: 2.298071537923439e-199 | mse:
0.7953874475264291
```

[42]:
```
y_predt = resultt.predict(Xt)
mset = mean_squared_error(yt, y_predt)

fstatt, pvaluet = sm.stats.linear_rainbow(resultt)
print(f' f stat: {fstatt} | p value: {pvaluet} | mse: {mset}')
```

```
 f stat: 1.0778265802478366 | p value: 4.45116439314793e-37 | mse:
0.000973979138532694
```

---

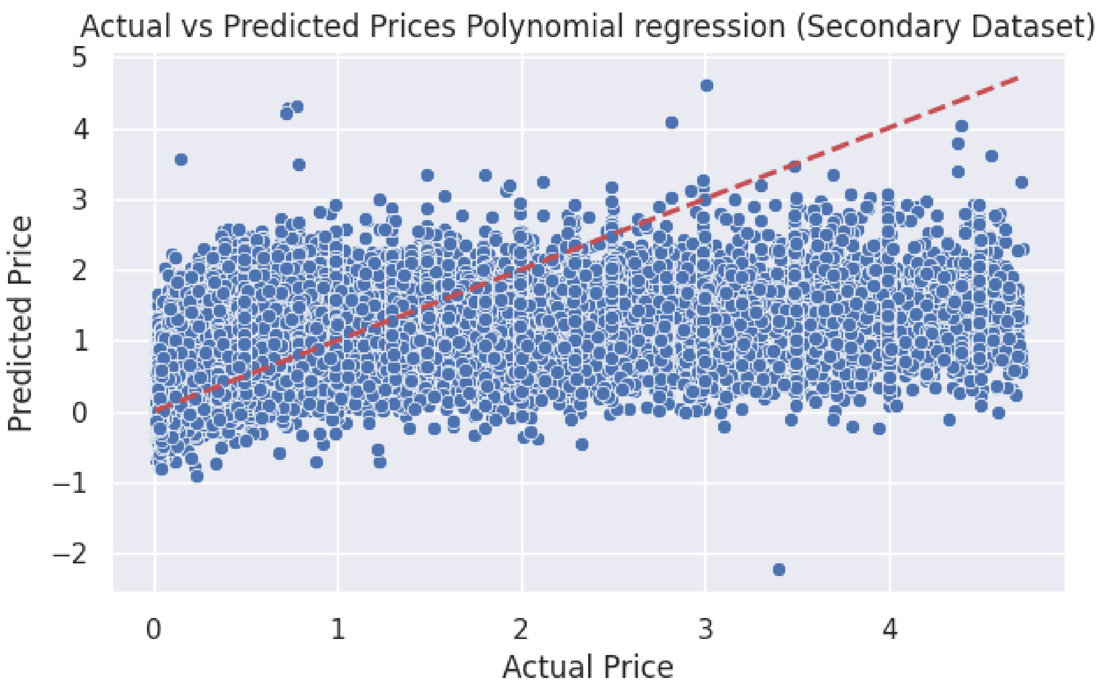
3 Polynomial Regression Prediction

[43]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Create linear regression model and use it based on selected feature and target
def prediction(feature, target):
    X_train, X_test, y_train, y_test = train_test_split(feature, target,␣
 ↪test_size=0.2, random_state=42)

    poly = PolynomialFeatures(degree=2)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)

    model = LinearRegression().fit(X_train_poly, y_train)

    y_pred = model.predict(X_test_poly)

    # Calculate metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return y_test, y_pred, mse, r2
```

[52]:
```
import statsmodels.api as sm
# Price Prediction based on Rank

X = data.loc[:, ~data.columns.isin(['price'])]
y = data["price"]

# Polinomial Linear Regression
y_test, y_pred, mse, r2 = prediction(X, y)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
# Ploting results

plt.figure(figsize=(7,4))
sns.scatterplot(x=y_test, y=y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
  ↪lw=2)
plt.title(f'Actual vs Predicted Prices Polynomial regression (Secondary␣
  ↪Dataset)')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.show()
```

```
Mean Squared Error: 0.673744456724405
R-squared: 0.33012267116035476
```

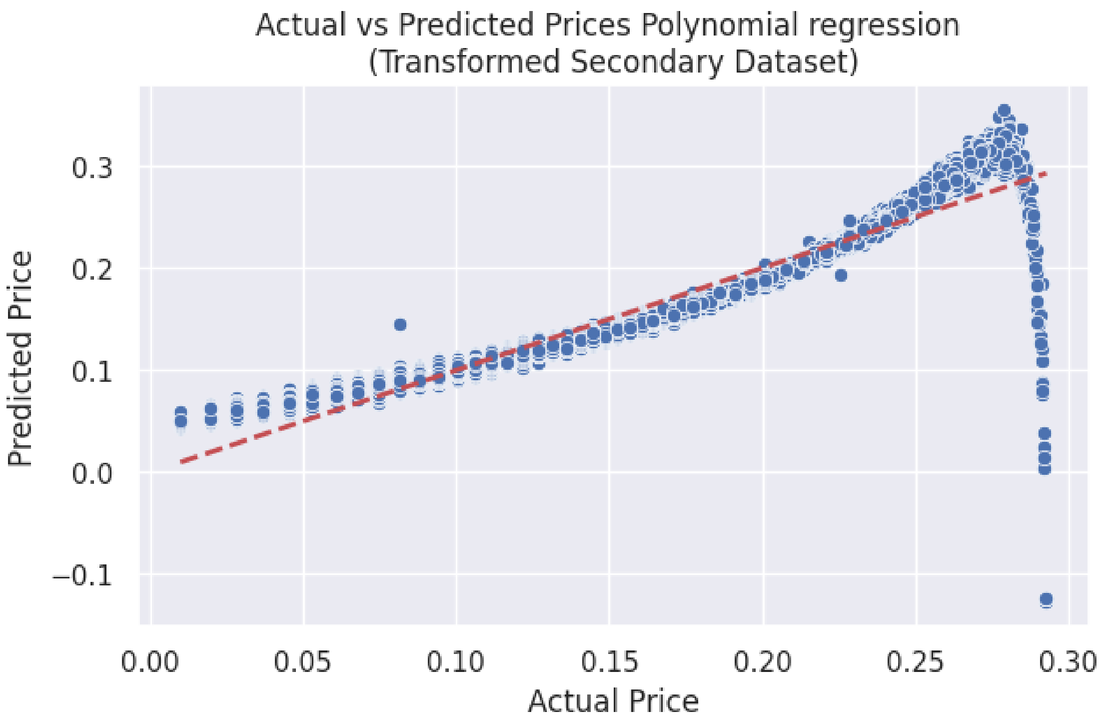


[51]:

```
Xt = transform.loc[:, ~transform.columns.isin(['price'])]
yt = transform["price"]

# Polinomial Linear Regression
y_testt, y_predt, mset, r2t = prediction(Xt, yt)

print(f'Mean Squared Error: {mset}')
print(f'R-squared: {r2t}')
```

```
# Ploting results

plt.figure(figsize=(7,4))
sns.scatterplot(x=y_testt, y=y_predt)
plt.plot([y_testt.min(), y_testt.max()], [y_testt.min(), y_testt.max()], 'r--',␣
↪lw=2)
plt.title(f'Actual vs Predicted Prices Polynomial regression \n(Transformed␣
↪Secondary Dataset)')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.show()
```

```
Mean Squared Error: 0.00022237500034004224
R-squared: 0.9447950218869814
```



4 Random Forest implementation

[46]:
```
# Random Forest implementation

# Modelling
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
↪recall_score, ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from scipy.stats import randint
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn import utils

# Tree Visualisation
from sklearn.tree import export_graphviz
from IPython.display import Image
import graphviz
```

[47]:
```
# define features
X = data.loc[:, ~data.columns.isin(['price'])]
#convert y values to categorical values
y = data["price"]


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


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


rf = RandomForestRegressor(n_estimators=100, random_state=42).
↪fit(X_train_scaled, y_train)
```

[48]:
```
# define features
Xt = transform.loc[:, ~transform.columns.isin(['price'])]
#convert y values to categorical values
yt = transform["price"]


# Split the data into training and test sets
X_traint, X_testt, y_traint, y_testt = train_test_split(Xt, yt, test_size=0.2)
```

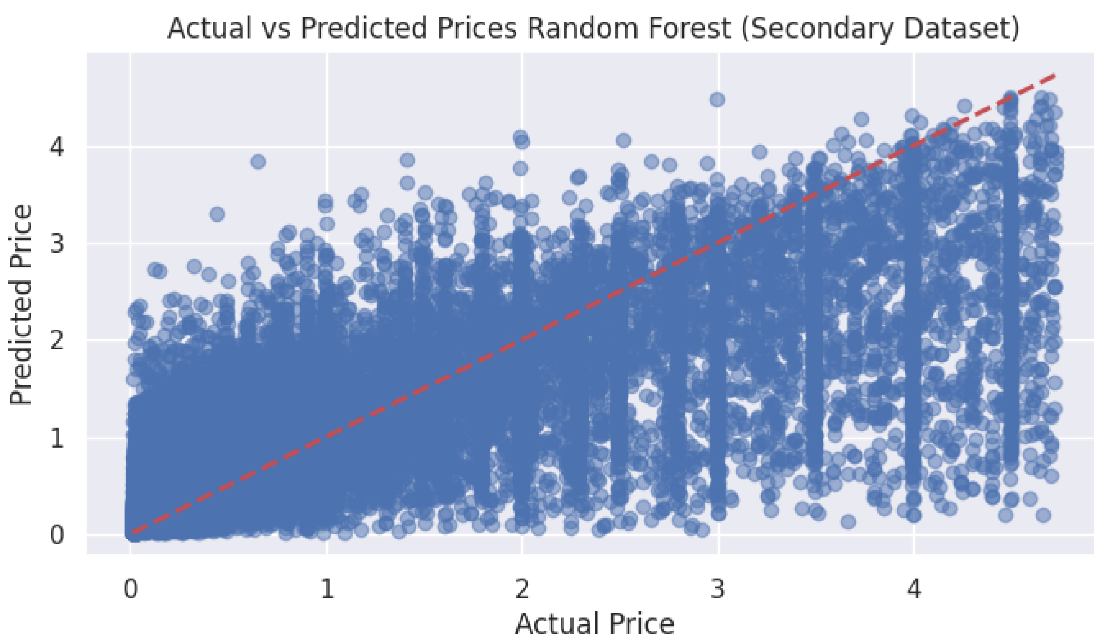
```
scalert = StandardScaler()
X_train_scaledt = scalert.fit_transform(X_traint)
X_test_scaledt = scalert.transform(X_testt)


rft = RandomForestRegressor(n_estimators=100, random_state=42).
 ↪fit(X_train_scaledt, y_traint)
```

[49]:

```
# Make predictions
y_pred = rf.predict(X_test_scaled)

# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Feature importance
feature_importance = pd.DataFrame({'feature': X.columns, 'importance': rf.
 ↪feature_importances_})
feature_importance = feature_importance.sort_values('importance',␣
 ↪ascending=False)
print(feature_importance)


# Visualize actual vs predicted prices
plt.figure(figsize=(8, 4))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪lw=2)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices Random Forest (Secondary Dataset)')
plt.show()
```

```
Mean Squared Error: 0.343129633959937
R-squared: 0.6601292746960944
              feature  importance
4          edhrecRank    0.181729
5     edhrecSaltiness    0.095045
19            setCode    0.079812
13             number    0.072983
18             rarity    0.072808
12               name    0.058863
0              artist    0.057624
16      priceProvider    0.049279
```

```
10           manaCost    0.047014
14       originalType    0.041473
1          cardFinish    0.037675
6    gameAvailability    0.035206
22               type    0.031632
17    providerListing    0.029222
11          manaValue    0.024442
2       colorIdentity    0.014953
3              colors    0.014196
21          toughness    0.012591
15              power    0.012520
7           isReprint    0.011894
23              types    0.010407
20         supertypes    0.004187
9              layout    0.003434
8            language    0.001012
```



[50]:
```
# Make predictions
y_predt = rft.predict(X_test_scaledt)

# Calculate metrics
mset = mean_squared_error(y_testt, y_predt)
r2t = r2_score(y_testt, y_predt)

print(f'Mean Squared Error: {mset}')
print(f'R-squared: {r2t}')
```

```python
# Feature importance
feature_importancet = pd.DataFrame({'feature': Xt.columns, 'importance': rft.
 ↪feature_importances_})
feature_importancet = feature_importancet.sort_values('importance',␣
 ↪ascending=False)
print(feature_importancet)


# Visualize actual vs predicted prices
plt.figure(figsize=(8, 4))
plt.scatter(y_testt, y_predt, alpha=0.5)
plt.plot([y_testt.min(), y_testt.max()], [y_testt.min(), y_testt.max()], 'r--',␣
 ↪lw=2)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices Random Forest \n(Transformed Secondary␣
 ↪Dataset)')
plt.show()
```

```
Mean Squared Error: 1.4471578909096415e-13
R-squared: 0.9999999999643058
             feature    importance
4    edhrecSaltiness  1.000000e+00
9           manaCost  3.874254e-09
3         edhrecRank  9.068164e-10
10         manaValue  9.032324e-10
11              name  8.535218e-10
21              type  5.706776e-10
22             types  4.802382e-10
2             colors  4.437794e-10
0             artist  3.695733e-10
13      originalType  3.214942e-10
17            rarity  1.791454e-10
20         toughness  1.755881e-10
19        supertypes  1.511267e-10
14             power  1.510587e-10
12            number  3.133510e-11
18           setCode  3.102427e-11
5   gameAvailability  8.727007e-12
8             layout  5.906089e-12
6          isReprint  2.962961e-12
15     priceProvider  2.027806e-12
1         cardFinish  1.565508e-12
16   providerListing  4.630754e-13
7           language  3.300786e-13
```


Actual vs Predicted Prices Random Forest
(Transformed Secondary Dataset)
Predicted Price
Actual Price
0.00
0.05
0.10
0.15
0.20
0.25
0.30