Outliers STAT Ship Group Project Final

February 6, 2024

1 Outliers STAT Ship Group Project Final

1.0.1 Import All Important Libraries

[64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from bioinfokit import visuz
from faker import Faker
import warnings
warnings.filterwarnings('ignore')
from sklearn.cluster import KMeans
import statsmodels.api as sm
import pylab as py
from sklearn.model_selection import cross_val_score, cross_val_predict
```

1.0.2 Importing Dataset and Defining Dataframes

[65]:
```
shipset = pd.read_excel("RegressionDataset.xlsx")
```

Exhibit 1

[66]:
```
shipset.rename(columns={"Age at Sale": "Age_at_Sale"},inplace = True) ; shipset
```

[66]:
```
     SaleDate              Vessel  Price  YearBuilt  Age_at_Sale    DWT  \
0  2022-01-07     Lowlands Beilun   73.0       1999            8  170.2
1  2022-01-07            CHS Moon   45.0       1991           16  150.2
2  2022-01-07        Spring Brave   62.0       1995           12  151.1
3  2022-01-07      Martha Verity   60.0       1995           12  158.0
4  2022-01-07             TMT TBN   61.3       1993           14  174.7
5  2022-02-07         Pantelis SP   83.0       1999            8  169.9
6  2022-02-07              Amazon   45.0       1990           17  149.5
```

7	2022-03-07	Cape Kassos	100.0	2004	3	170.0
8	2022-03-07	Johnny K	65.0	1994	13	165.3
9	2022-03-07	Zorbas	70.0	1996	11	165.1
10	2022-03-07	Americana	33.0	1987	20	149.0
11	2022-03-07	Martha Verity	63.0	1995	12	158.0
12	2022-03-07	Ullswater	43.0	1990	17	123.5
13	2022-04-07	Formosabulk Brave	95.0	2001	6	170.1
14	2022-04-07	Formosabulk Clement	95.0	2001	6	170.1
15	2022-04-07	Nautical Dream	63.5	1994	13	151.4
16	2022-04-07	Formosabulk Allstart	67.0	1995	12	150.4
17	2022-04-07	Arimathian	62.0	1994	13	149.8
18	2022-04-07	Boss	31.0	1985	22	139.8
19	2022-05-07	Zorbas II	86.0	1996	11	174.5
20	2022-05-07	Fertilia	50.5	1997	10	172.6
21	2022-05-07	Ingenious	64.2	1996	11	170.0
22	2022-06-07	Anangel Dawn	67.0	1994	13	149.3
23	2022-06-07	Orient Fortune	28.0	1984	23	161.4
24	2022-07-07	Great Moon	30.0	1984	23	146.0
25	2022-07-07	Gran Trader	105.0	2001	6	172.5
26	2022-08-07	Cape Brazil	22.0	1981	26	140.8
27	2022-09-07	Thalassini Kyra	133.0	2002	5	164.2
28	2022-10-07	Tiger Lily	90.0	1995	12	149.2
29	2022-10-07	Dong-A-Helios	47.0	1986	21	146.9
30	2022-10-07	Marine Hunter	45.0	1984	23	164.5
31	2022-10-07	Peace Glory	57.0	1984	23	166.1
32	2022-11-07	Sumihou	106.0	1996	11	171.1
33	2022-11-07	Gran Trader	152.0	2001	6	172.6
34	2022-11-07	Netadola	97.0	1993	14	149.5
35	2022-11-07	Nordstar	38.0	1983	24	150.7
36	2022-11-07	Captain Vangelis L	87.5	1992	15	148.2
37	2022-12-07	Voutakos	78.0	1987	20	188.3
38	2022-12-07	Sachuest	35.0	1986	21	98.4
39	2022-01-08	Sinfonia	83.7	1991	17	184.4
40	2022-01-08	Jin Tai	155.0	2004	4	173.9
41	2022-02-08	Dias	58.0	1988	20	135.0
42	2022-03-08	Desimi	83.0	1989	19	207.1
43	2022-03-08	Samos	25.0	1982	26	137.0
44	2022-03-08	Cape Sun	135.0	1999	9	171.7
45	2022-04-08	Nightflight	158.0	2004	4	170.0
46	2022-05-08	Cape Falcon	87.2	1993	15	161.5
47	2022-05-08	Castle Peak	82.0	1990	18	145.4

	Capesize	Month
0	4647	1
1	4647	1
2	4647	1
3	4647	1

4	4647	1
5	4878	2
6	4878	2
7	5245	3
8	5245	3
9	5245	3
10	5245	3
11	5245	3
12	5245	3
13	5752	4
14	5752	4
15	5752	4
16	5752	4
17	5752	4
18	5752	4
19	6201	5
20	6201	5
21	6201	5
22	6618	6
23	6618	6
24	6980	7
25	6980	7
26	7441	8
27	8181	9
28	8886	10
29	8886	10
30	8886	10
31	8886	10
32	9663	11
33	9663	11
34	9663	11
35	9663	11
36	9663	11
37	10299	12
38	10299	12
39	10526	1
40	10526	1
41	10844	2
42	11193	3
43	11193	3
44	11193	3
45	11614	4
46	12479	5
47	12479	5

1.0.3 Mean, STD, and other Metrics for the variables

Exhibit 2

[67]:
```
shipset.describe()
```

[67]:

	Price	YearBuilt	Age_at_Sale	DWT	Capesize
count	48.00000	48.000000	48.000000	48.000000	48.000000
mean	72.95625	1992.916667	14.270833	158.935417	7643.708333
std	33.89537	6.330720	6.330405	17.650984	2499.309368
min	22.00000	1981.000000	3.000000	98.400000	4647.000000
25%	46.50000	1987.750000	10.750000	149.275000	5245.000000
50%	66.00000	1994.000000	13.000000	161.450000	6799.000000
75%	88.12500	1996.250000	20.000000	170.125000	9663.000000
max	158.00000	2004.000000	26.000000	207.100000	12479.000000

	Month
count	48.000000
mean	5.312500
std	3.543987
min	1.000000
25%	3.000000
50%	4.000000
75%	8.250000
max	12.000000

1.0.4 Distributions of Variables in the dataset

Exhibit 3

[68]:
```
sns.distplot(shipset['Price'], color = 'pink')
plt.title("Price Distribution")
```

[68]: Text(0.5, 1.0, 'Price Distribution')

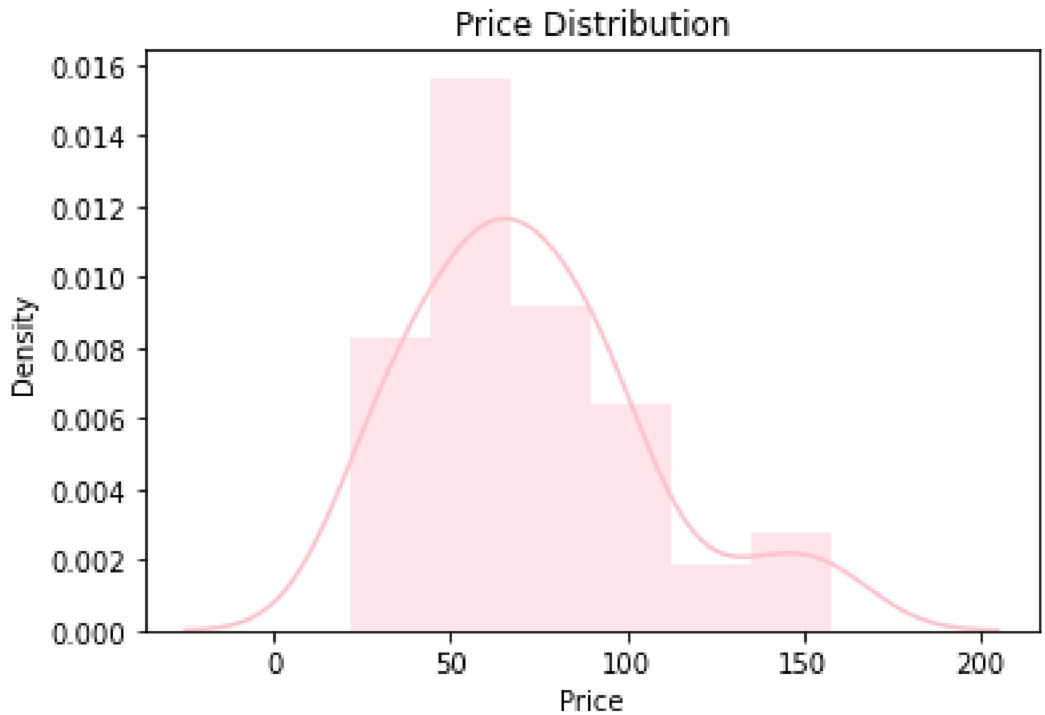


Exhibit 4

[69]:
```
sns.distplot(shipset['DWT'], color = 'blue')
plt.title("DWT Distribution")
```

[69]: Text(0.5, 1.0, 'DWT Distribution')

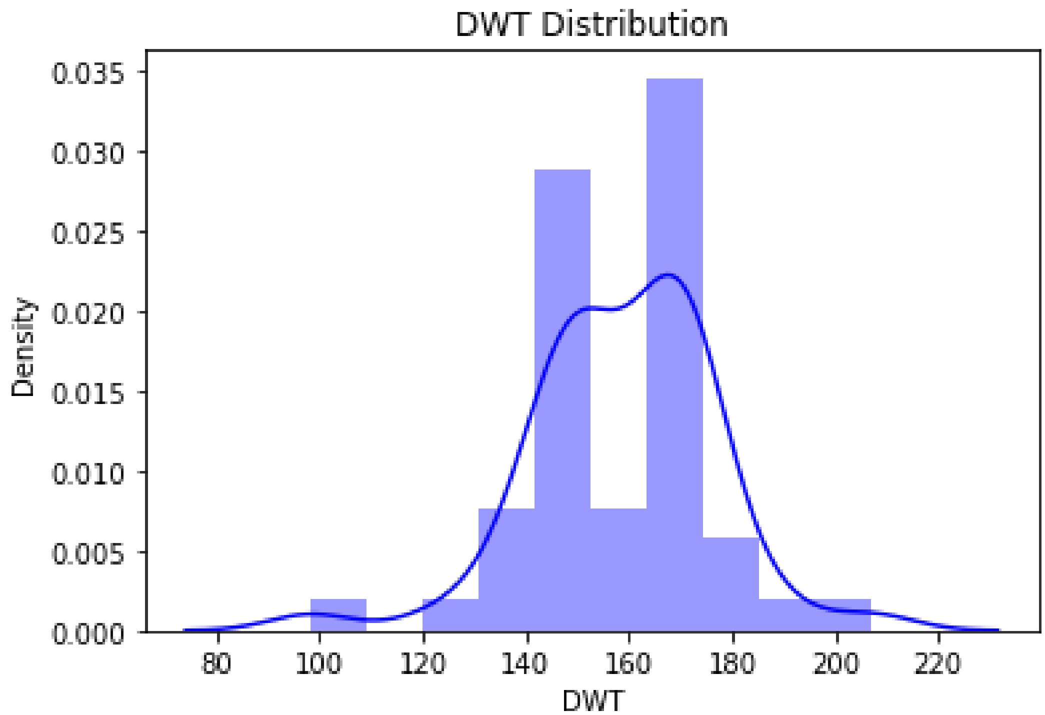


Exhibit 5

[70]:
```
sns.distplot(shipset['Age_at_Sale'], color = 'black')
plt.title("Age at Sale Distribution")
```

[70]: Text(0.5, 1.0, 'Age at Sale Distribution')

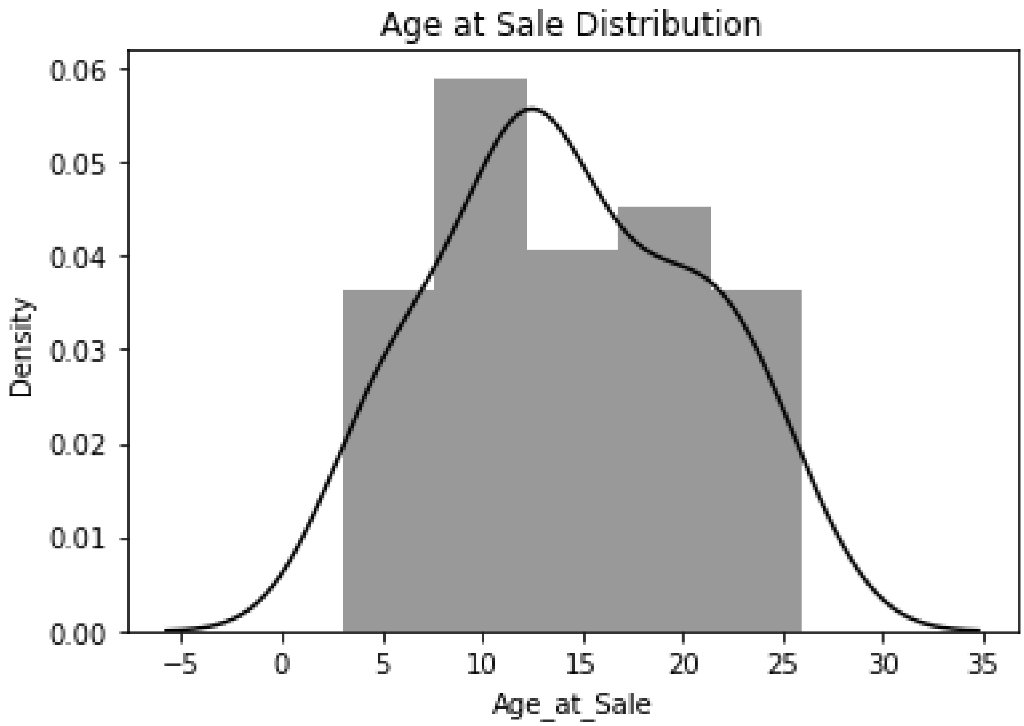
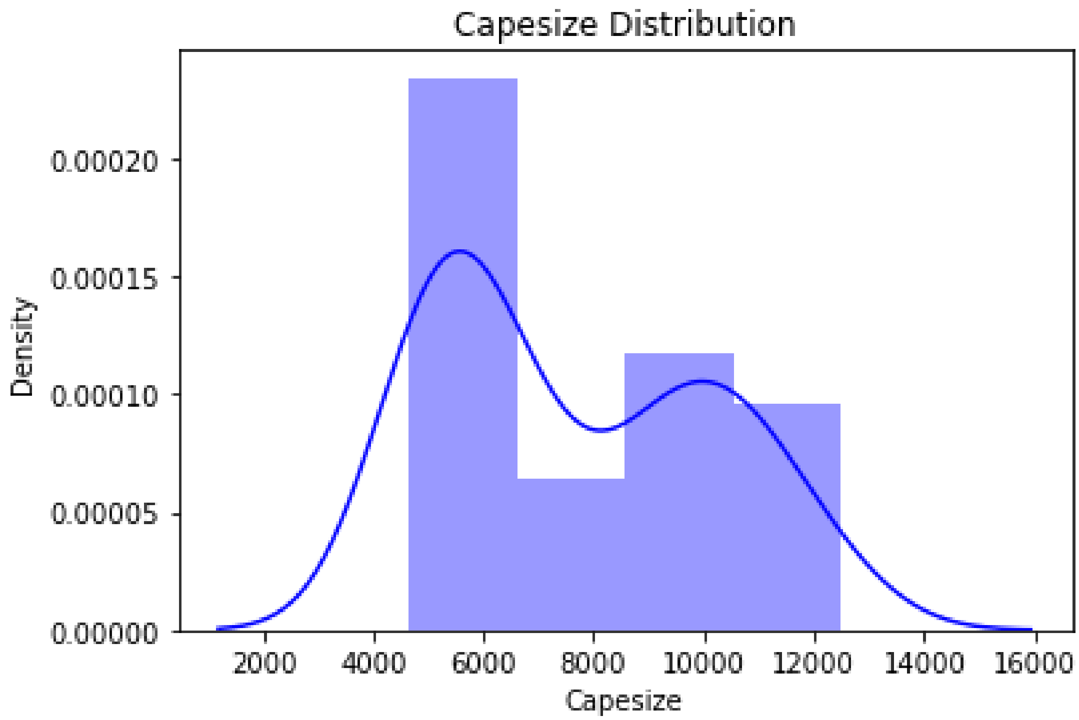


Exhibit 6

[71]:
```
sns.distplot(shipset['Capesize'], color = 'blue')
plt.title("Capesize Distribution")
```

[71]: Text(0.5, 1.0, 'Capesize Distribution')



1.0.5 Relations of Variables in the dataset

Exhibit 7

[72]:
```
sns.regplot(x = shipset['Age_at_Sale'], y = shipset['Price'],marker = 'o',␣
 ↪color = 'grey')
plt.title('Age_at_Sale vs Price')
```

[72]:
```
Text(0.5, 1.0, 'Age_at_Sale vs Price')
```

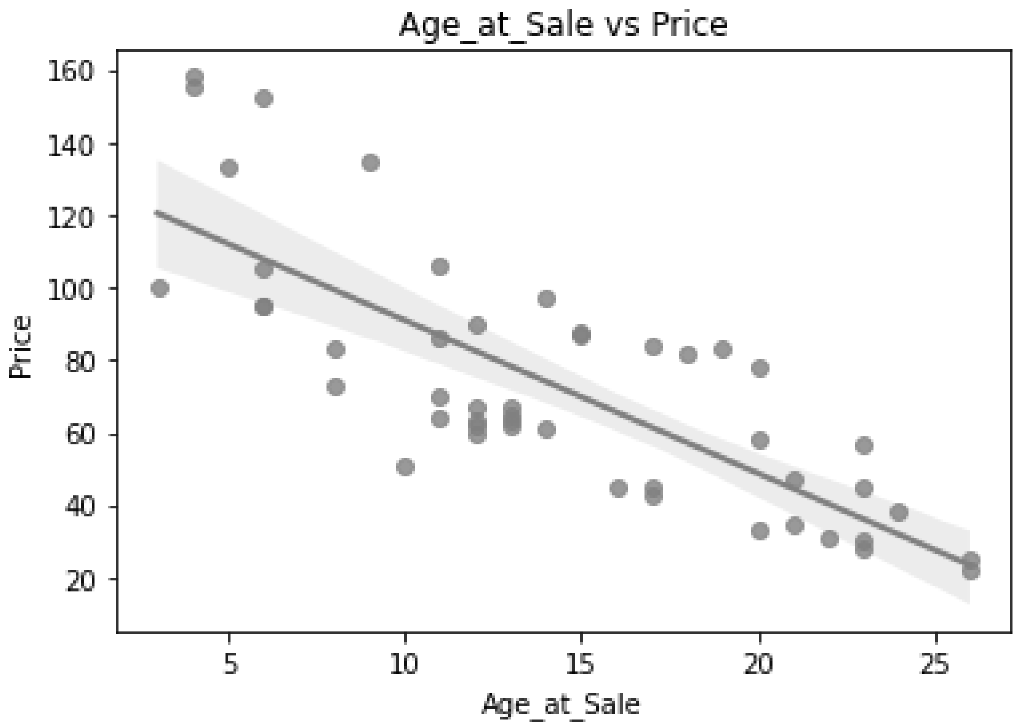


Exhibit 8

[73]:
```
sns.regplot(x = shipset['Capesize'], y = shipset['Price'],marker = '*', color =
 ↪'orange')
plt.title('Capesize vs Price')
```

[73]: Text(0.5, 1.0, 'Capesize vs Price')

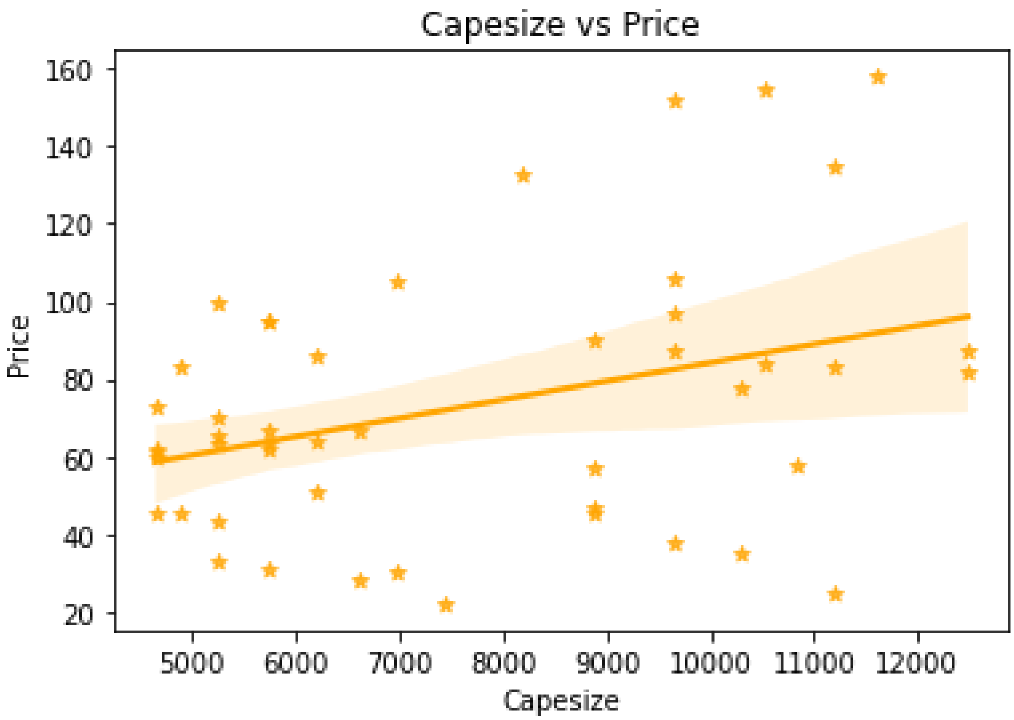


Exhibit 9

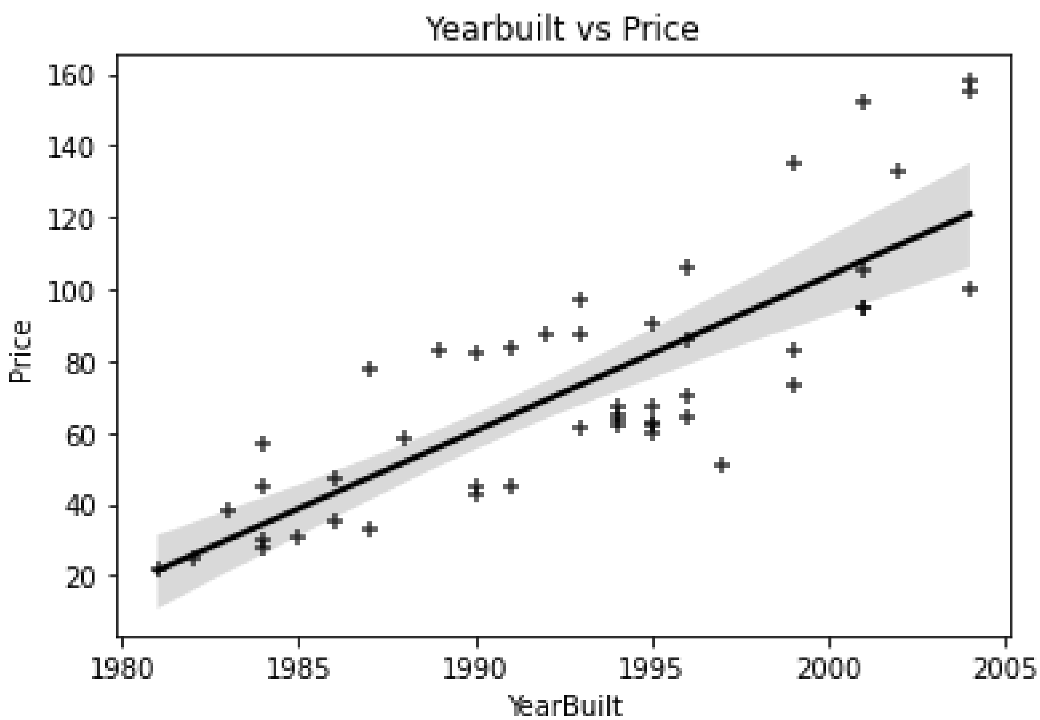
[74]:
```
sns.regplot(x = shipset['DWT'], y = shipset['Price'],marker = '*', color =␣
↪'red')
plt.title('DWT vs Price')
```

[74]:
```
Text(0.5, 1.0, 'DWT vs Price')
```

Exhibit 10

[75]:
```
sns.regplot(x = shipset['YearBuilt'], y = shipset['Price'],marker = '+', color␣
↪= 'black')
plt.title('Yearbuilt vs Price')
```

[75]: Text(0.5, 1.0, 'Yearbuilt vs Price')



1.1 Bet Performer Identification

Exhibit 11: Using Euclidean Distance Performed on Excel by Arunabh Choudhury

[76]:
```
BetPerformer = pd.read_excel("s.xlsx","final") ; BetPerformer.sort_values(by =
↪['Euclidean Distance'], ascending = True).head(1)
```

[76]:

	Vessel	Price	Euclidean Distance	Manhattan Distance
0	Cape Sun	135.0	0.185856	0.253952

1.1.1 Correlation and VIF

Exhibit 12

[77]:
```
shipset.corr()
```

[77]:

	Price	YearBuilt	Age_at_Sale	DWT	Capesize	Month
Price	1.000000	0.808303	-0.787491	0.514805	0.352348	0.037932
YearBuilt	0.808303	1.000000	-0.998059	0.441826	-0.172633	-0.282364
Age_at_Sale	-0.787491	-0.998059	1.000000	-0.431264	0.217360	0.262640
DWT	0.514805	0.441826	-0.431264	1.000000	0.042766	-0.160653
Capesize	0.352348	-0.172633	0.217360	0.042766	1.000000	0.427984
Month	0.037932	-0.282364	0.262640	-0.160653	0.427984	1.000000

[78]:
```
fig,ax = plt.subplots(figsize=(8,6))
sns.heatmap(shipset.corr(),vmin=-0.8, annot=True, cmap='Greys',ax=ax);
```

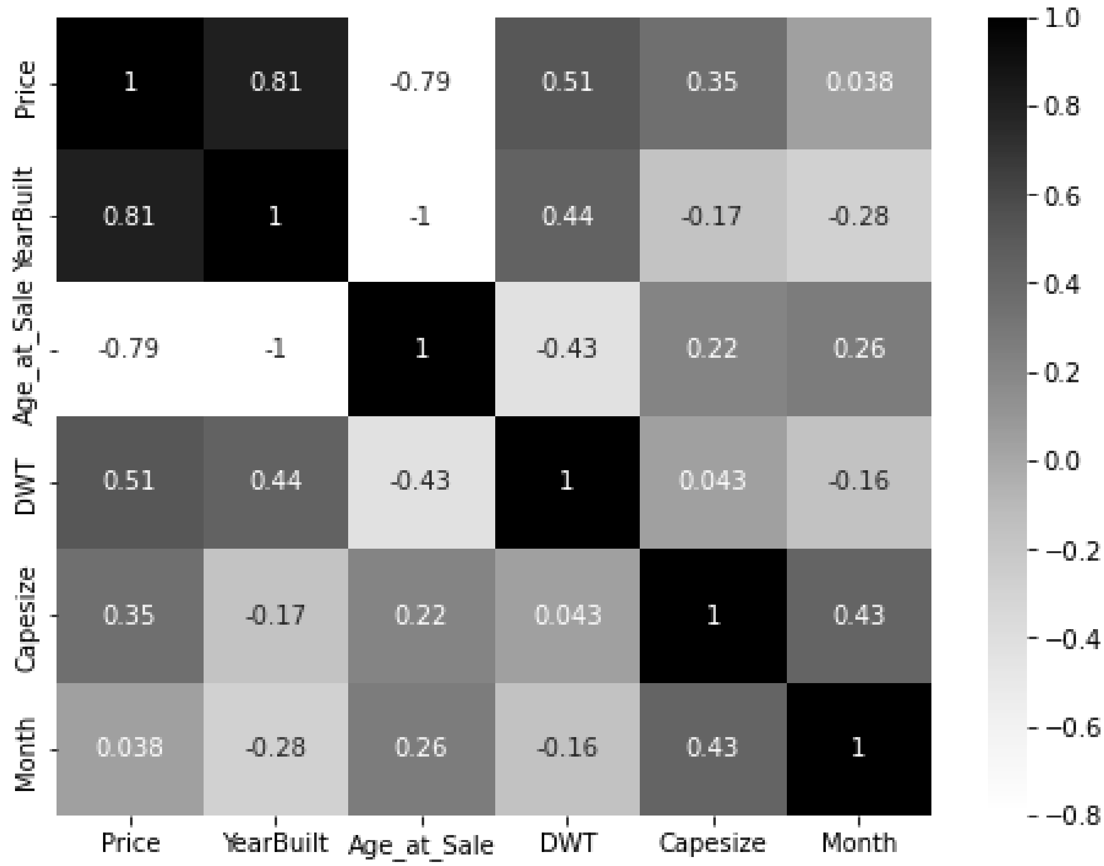


Exhibit 13

[79]:
```
Y, X = dmatrices('Price ~ DWT+Age_at_Sale+Capesize', data=shipset,␣
↪return_type='dataframe')
```

[80]:
```
vif = pd.DataFrame()
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['variable'] = X.columns ; vif
```

[80]:

	VIF	variable
0	131.944976	Intercept
1	1.258727	DWT
2	1.318729	Age_at_Sale
3	1.075427	Capesize

1.2 Linear Regression (without Standardization)

[81]:
```
shipset_regression = shipset[["Age_at_Sale","DWT","Capesize","Price"]].copy() ;
↪shipset_regression_results =
↪shipset[["Age_at_Sale","DWT","Capesize","Price"]].copy()
```

[82]:
```
X = shipset_regression.values[:,:-1] ; Y = shipset_regression.values[:,-1]
```

[83]:
```
LR1 = LinearRegression().fit(X,Y)
```

[84]:
```
m = LR1.coef_.flatten() ; b = LR1.intercept_.flatten()
```

[85]:
```
print("m = {0}".format(m)) ; print("b = {0}".format(b))
```

```
m = [-4.54380392  0.24215462  0.00720692]
b = [44.22554998]
```

[86]:
```
LR1_Y_predicted = LR1.predict(X)
```

[87]:
```
shipset_regression_results['LR1_Y_predicted'] = LR1_Y_predicted
```

[88]:
```
shipset_regression_results
```

[88]:

	Age_at_Sale	DWT	Capesize	Price	LR1_Y_predicted
0	8	170.2	4647	73.0	82.580410
1	16	150.2	4647	45.0	41.386886
2	12	151.1	4647	62.0	59.780041
3	12	158.0	4647	60.0	61.450908
4	14	174.7	4647	61.3	56.407282
5	8	169.9	4878	83.0	84.172563
6	17	149.5	4878	45.0	38.338373
7	3	170.0	5245	100.0	109.560739
8	13	165.3	5245	65.0	62.984573
9	11	165.1	5245	70.0	72.023750
10	20	149.0	5245	33.0	27.230825
11	12	158.0	5245	63.0	65.760648
12	17	123.5	5245	43.0	34.687294
13	6	170.1	5752	95.0	99.607453
14	6	170.1	5752	95.0	99.607453
15	13	151.4	5752	63.5	63.272534
16	12	150.4	5752	67.0	67.574183
17	13	149.8	5752	62.0	62.885087
18	22	139.8	5752	31.0	19.569305
19	11	174.5	6201	86.0	81.189822
20	10	172.6	6201	50.5	85.273532
21	11	170.0	6201	64.2	80.100126
22	13	149.3	6618	67.0	69.005205
23	23	161.4	6618	28.0	26.497237

24	23	146.0	6980	30.0	25.376962
25	6	172.5	6980	105.0	109.038726
26	26	140.8	7441	22.0	13.808738
27	5	164.2	8181	133.0	120.228162
28	12	149.2	8886	90.0	89.870096
29	21	146.9	8886	47.0	48.418906
30	23	164.5	8886	45.0	43.593219
31	23	166.1	8886	57.0	43.980666
32	11	171.1	9663	106.0	105.316866
33	6	172.6	9663	152.0	128.399118
34	14	149.5	9663	97.0	86.454915
35	24	150.7	9663	38.0	41.307461
36	15	148.2	9663	87.5	81.596310
37	20	188.3	10299	78.0	73.171294
38	21	98.4	10299	35.0	46.857790
39	17	184.4	10526	83.7	87.494274
40	4	173.9	10526	155.0	144.021102
41	20	135.0	10844	58.0	64.192226
42	19	207.1	11193	83.0	88.710595
43	26	137.0	11193	25.0	39.928928
44	9	171.7	11193	135.0	125.576360
45	4	170.0	11614	158.0	150.917832
46	15	161.5	12479	87.2	105.111663
47	18	145.4	12479	82.0	87.581562

[89]:
```
residuals_LR1 = Y - LR1_Y_predicted
```

[90]:
```
shipset_regression_results['LR1 Residuals'] = residuals_LR1
```

[91]:
```
r_squared_LR1 =  LR1.score(X, Y) ; print(r_squared_LR1)
```

```
0.9204352585883622
```

[92]:
```
print(mean_absolute_error(Y, LR1_Y_predicted))
```

```
6.860174471941046
```

[93]:
```
print(mean_squared_error(Y, LR1_Y_predicted))
```

```
89.50721469438948
```

1.2.1 Residual Plot for Linear Regression (without Standardization)

Exhibit 14

[94]:
```
sns.scatterplot(data = shipset_regression_results, x = 'LR1_Y_predicted' , y = 'LR1 Residuals', color = 'green')
plt.title("Residual vs Predicted Plot for Linear Regression (without Standardization)")
plt.axhline(y=0)
```

```
plt.ylim(-75,75)
```

[94]: (-75.0, 75.0)

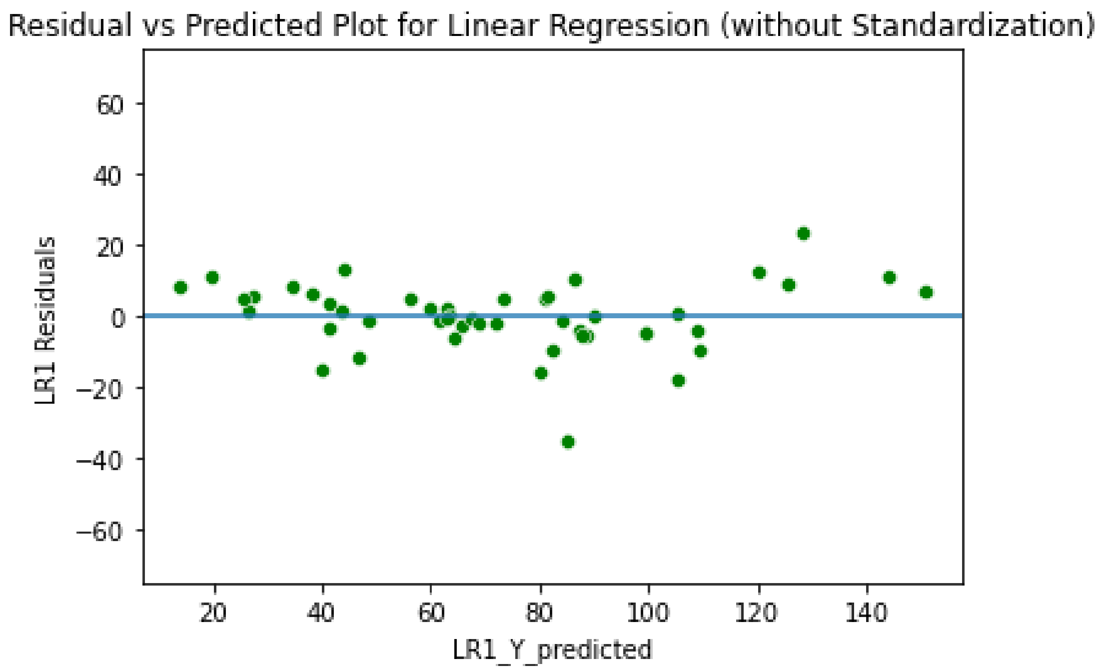
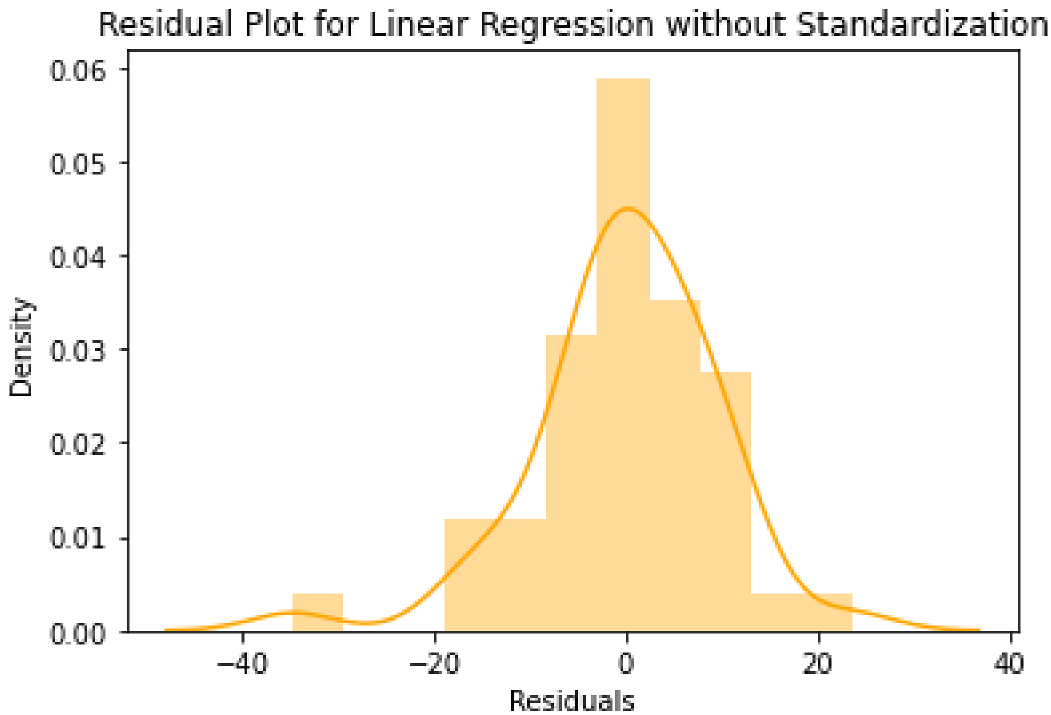


Exhibit 15

[95]:
```
sns.distplot(residuals_LR1, color = 'orange')
plt.title("Residual Plot for Linear Regression without Standardization")
plt.xlabel("Residuals")
```

[95]: Text(0.5, 0, 'Residuals')



1.3 Cross Validation of Linear Regression without Standardization

[96]:
```
scores = cross_val_score(LR1, X, Y, cv=5)
print ("Cross-validated scores:", scores.mean())
```

```
Cross-validated scores: 0.8715531823056342
```

1.4 OLS on Non Standardized Dataset

[97]:
```
X = sm.add_constant(X)
```

[98]:
```
ols1 = sm.OLS(Y, X).fit()
```

[99]:
```
ols1_predictions = ols1.predict(X)
```

[100]:
```
shipset_regression_results['OLS1_Y_predicted'] = ols1_predictions
```

[101]:
```
shipset_regression_results['OLS1_Residuals'] = ols1.resid ; print(ols1.resid.
 ↪mean())
```

```
4.794979228487743e-12
```

[102]:
```
print(mean_absolute_error(Y, ols1_predictions))
```

```
6.860174471941328
```

[103]:
```
print(mean_squared_error(Y, ols1_predictions))
```

89.50721469438948

[104]:
```
shipset_regression_results
```

[104]:

	Age_at_Sale	DWT	Capesize	Price	LR1_Y_predicted	LR1 Residuals	\
0	8	170.2	4647	73.0	82.580410	-9.580410	
1	16	150.2	4647	45.0	41.386886	3.613114	
2	12	151.1	4647	62.0	59.780041	2.219959	
3	12	158.0	4647	60.0	61.450908	-1.450908	
4	14	174.7	4647	61.3	56.407282	4.892718	
5	8	169.9	4878	83.0	84.172563	-1.172563	
6	17	149.5	4878	45.0	38.338373	6.661627	
7	3	170.0	5245	100.0	109.560739	-9.560739	
8	13	165.3	5245	65.0	62.984573	2.015427	
9	11	165.1	5245	70.0	72.023750	-2.023750	
10	20	149.0	5245	33.0	27.230825	5.769175	
11	12	158.0	5245	63.0	65.760648	-2.760648	
12	17	123.5	5245	43.0	34.687294	8.312706	
13	6	170.1	5752	95.0	99.607453	-4.607453	
14	6	170.1	5752	95.0	99.607453	-4.607453	
15	13	151.4	5752	63.5	63.272534	0.227466	
16	12	150.4	5752	67.0	67.574183	-0.574183	
17	13	149.8	5752	62.0	62.885087	-0.885087	
18	22	139.8	5752	31.0	19.569305	11.430695	
19	11	174.5	6201	86.0	81.189822	4.810178	
20	10	172.6	6201	50.5	85.273532	-34.773532	
21	11	170.0	6201	64.2	80.100126	-15.900126	
22	13	149.3	6618	67.0	69.005205	-2.005205	
23	23	161.4	6618	28.0	26.497237	1.502763	
24	23	146.0	6980	30.0	25.376962	4.623038	
25	6	172.5	6980	105.0	109.038726	-4.038726	
26	26	140.8	7441	22.0	13.808738	8.191262	
27	5	164.2	8181	133.0	120.228162	12.771838	
28	12	149.2	8886	90.0	89.870096	0.129904	
29	21	146.9	8886	47.0	48.418906	-1.418906	
30	23	164.5	8886	45.0	43.593219	1.406781	
31	23	166.1	8886	57.0	43.980666	13.019334	
32	11	171.1	9663	106.0	105.316866	0.683134	
33	6	172.6	9663	152.0	128.399118	23.600882	
34	14	149.5	9663	97.0	86.454915	10.545085	
35	24	150.7	9663	38.0	41.307461	-3.307461	
36	15	148.2	9663	87.5	81.596310	5.903690	
37	20	188.3	10299	78.0	73.171294	4.828706	
38	21	98.4	10299	35.0	46.857790	-11.857790	
39	17	184.4	10526	83.7	87.494274	-3.794274	
40	4	173.9	10526	155.0	144.021102	10.978898	

41	20	135.0	10844	58.0	64.192226	-6.192226
42	19	207.1	11193	83.0	88.710595	-5.710595
43	26	137.0	11193	25.0	39.928928	-14.928928
44	9	171.7	11193	135.0	125.576360	9.423640
45	4	170.0	11614	158.0	150.917832	7.082168
46	15	161.5	12479	87.2	105.111663	-17.911663
47	18	145.4	12479	82.0	87.581562	-5.581562

	OLS1_Y_predicted	OLS1_Residuals
0	82.580410	-9.580410
1	41.386886	3.613114
2	59.780041	2.219959
3	61.450908	-1.450908
4	56.407282	4.892718
5	84.172563	-1.172563
6	38.338373	6.661627
7	109.560739	-9.560739
8	62.984573	2.015427
9	72.023750	-2.023750
10	27.230825	5.769175
11	65.760648	-2.760648
12	34.687294	8.312706
13	99.607453	-4.607453
14	99.607453	-4.607453
15	63.272534	0.227466
16	67.574183	-0.574183
17	62.885087	-0.885087
18	19.569305	11.430695
19	81.189822	4.810178
20	85.273532	-34.773532
21	80.100126	-15.900126
22	69.005205	-2.005205
23	26.497237	1.502763
24	25.376962	4.623038
25	109.038726	-4.038726
26	13.808738	8.191262
27	120.228162	12.771838
28	89.870096	0.129904
29	48.418906	-1.418906
30	43.593219	1.406781
31	43.980666	13.019334
32	105.316866	0.683134
33	128.399118	23.600882
34	86.454915	10.545085
35	41.307461	-3.307461
36	81.596310	5.903690
37	73.171294	4.828706

```
38        46.857790     -11.857790
39        87.494274      -3.794274
40       144.021102      10.978898
41        64.192226      -6.192226
42        88.710595      -5.710595
43        39.928928     -14.928928
44       125.576360       9.423640
45       150.917832       7.082168
46       105.111663     -17.911663
47        87.581562      -5.581562
```

Exhibit 16

[105]:
```
ols1.summary()
```

[105]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.920
Model:                            OLS   Adj. R-squared:                  0.915
Method:                 Least Squares   F-statistic:                     169.7
Date:                Thu, 10 Nov 2022   Prob (F-statistic):           3.39e-24
Time:                        12:37:33   Log-Likelihood:                -175.97
No. Observations:                  48   AIC:                             359.9
Df Residuals:                      44   BIC:                             367.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         44.2255     16.383      2.699      0.010      11.207      77.244
x1            -4.5438      0.261    -17.378      0.000      -5.071      -4.017
x2             0.2422      0.092      2.643      0.011       0.058       0.427
x3             0.0072      0.001     12.051      0.000       0.006       0.008
==============================================================================
Omnibus:                       13.373   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               19.393
Skew:                          -0.851   Prob(JB):                     6.15e-05
Kurtosis:                       5.607   Cond. No.                     9.23e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.23e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

1.5 Linear Regression Equation

1.6 Price = Age_at_Sale*(-4.54) + DWT*(0.242) + Capesize*(0.00720) + 44.22

[106]:
```
DWT_custom = float(input('What is the ship\'s DWT \n')) ;
```

```
What is the ship's DWT
172
```

[107]:
```
Age_at_Sale_custom = int(input('What is the ship\'s age during sale \n')) ;
```

```
What is the ship's age during sale
11
```

[108]:
```
Capesize_index = float(input('What is the ship\'s capesize \n')) ;
```

```
What is the ship's capesize
12479
```

[109]:
```
Price_estimated =  DWT_custom*(0.242) + Capesize_index*(0.00720) +␣
↪Age_at_Sale_custom*(-4.54) + 44.22 ; print("\nEstimated Price of the ship is␣
↪= {0}".format(Price_estimated))
```

```

Estimated Price of the ship is = 125.75280000000001
```

2 OLS on Standardized Dataset

[110]:
```
shipset_standard = shipset[["Age_at_Sale","DWT","Capesize","Price"]].copy()
```

[111]:
```
shipset_standard=(shipset_standard-shipset_standard.min())/(shipset_standard.
↪max()-shipset_standard.min())
```

[112]:
```
shipset_standard
```

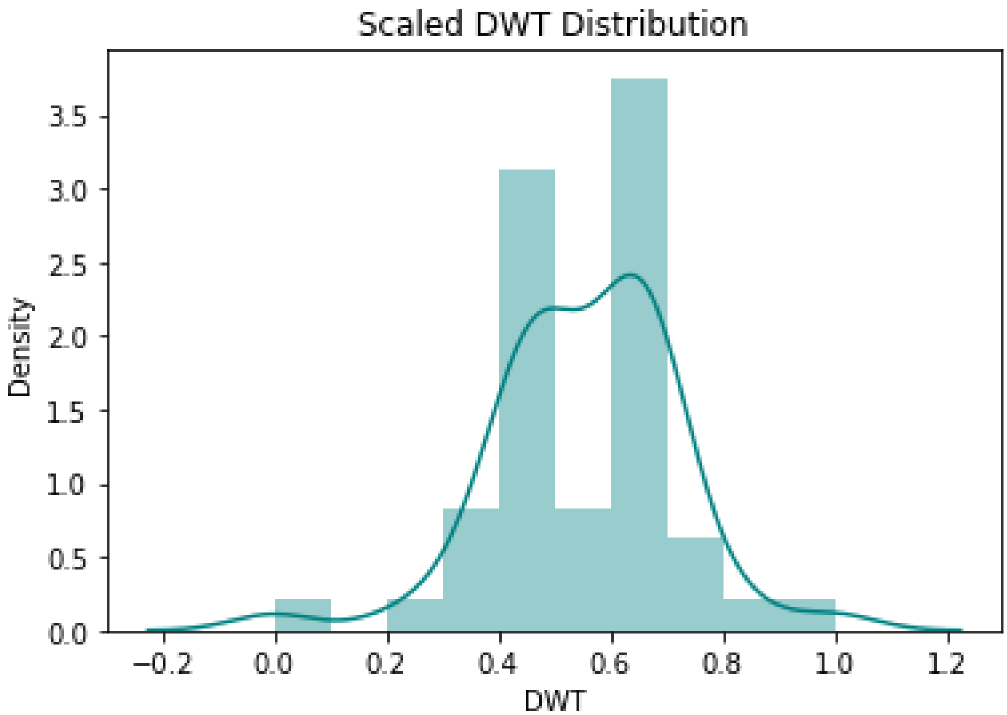
[112]:

	Age_at_Sale	DWT	Capesize	Price
0	0.217391	0.660534	0.000000	0.375000
1	0.565217	0.476541	0.000000	0.169118
2	0.391304	0.484821	0.000000	0.294118
3	0.391304	0.548298	0.000000	0.279412
4	0.478261	0.701932	0.000000	0.288971
5	0.217391	0.657774	0.029494	0.448529
6	0.608696	0.470101	0.029494	0.169118
7	0.000000	0.658694	0.076353	0.573529
8	0.434783	0.615455	0.076353	0.316176
9	0.347826	0.613615	0.076353	0.352941
10	0.739130	0.465501	0.076353	0.080882
11	0.391304	0.548298	0.076353	0.301471
12	0.608696	0.230911	0.076353	0.154412

13	0.130435	0.659614	0.141088	0.536765
14	0.130435	0.659614	0.141088	0.536765
15	0.434783	0.487580	0.141088	0.305147
16	0.391304	0.478381	0.141088	0.330882
17	0.434783	0.472861	0.141088	0.294118
18	0.826087	0.380865	0.141088	0.066176
19	0.347826	0.700092	0.198417	0.470588
20	0.304348	0.682613	0.198417	0.209559
21	0.347826	0.658694	0.198417	0.310294
22	0.434783	0.468261	0.251660	0.330882
23	0.869565	0.579577	0.251660	0.044118
24	0.869565	0.437902	0.297880	0.058824
25	0.130435	0.681693	0.297880	0.610294
26	1.000000	0.390064	0.356742	0.000000
27	0.086957	0.605336	0.451226	0.816176
28	0.391304	0.467341	0.541241	0.500000
29	0.782609	0.446182	0.541241	0.183824
30	0.869565	0.608096	0.541241	0.169118
31	0.869565	0.622815	0.541241	0.257353
32	0.347826	0.668813	0.640449	0.617647
33	0.130435	0.682613	0.640449	0.955882
34	0.478261	0.470101	0.640449	0.551471
35	0.913043	0.481141	0.640449	0.117647
36	0.521739	0.458142	0.640449	0.481618
37	0.739130	0.827047	0.721655	0.411765
38	0.782609	0.000000	0.721655	0.095588
39	0.608696	0.791168	0.750638	0.453676
40	0.043478	0.694572	0.750638	0.977941
41	0.739130	0.336707	0.791241	0.264706
42	0.695652	1.000000	0.835802	0.448529
43	1.000000	0.355106	0.835802	0.022059
44	0.260870	0.674333	0.835802	0.830882
45	0.043478	0.658694	0.889556	1.000000
46	0.521739	0.580497	1.000000	0.479412
47	0.652174	0.432383	1.000000	0.441176

[113]:
```
sns.distplot(shipset_standard['DWT'], color = 'teal')
plt.title("Scaled DWT Distribution")
```

[113]:
```
Text(0.5, 1.0, 'Scaled DWT Distribution')
```



[114]:
```
XS = shipset_standard.values[:,:-1] ; Y = shipset_standard.values[:,-1]
```

[115]:
```
XS = sm.add_constant(XS)
```

[116]:
```
ols2 = sm.OLS(Y, XS).fit()
```

[117]:
```
ols2_predictions = ols2.predict(XS)
```

[118]:
```
ols2_residuals = ols2.resid
```

[119]:
```
shipset_regression_results['OLS2_Residuals'] = ols2_residuals
```

[120]:
```
shipset_regression_results['OLS2_Y_predicted'] = ols2_predictions
```

[121]:
```
shipset_regression_results['OLS2_Y_predicted'] =␣
↪shipset_regression_results['OLS2_Y_predicted']*136 + 22
```

[122]:
```
shipset_regression_results
```

[122]:
```
   Age_at_Sale    DWT  Capesize  Price  LR1_Y_predicted  LR1 Residuals  \
0            8  170.2      4647   73.0        82.580410      -9.580410
1           16  150.2      4647   45.0        41.386886       3.613114
2           12  151.1      4647   62.0        59.780041       2.219959
3           12  158.0      4647   60.0        61.450908      -1.450908
```

4	14	174.7	4647	61.3	56.407282	4.892718
5	8	169.9	4878	83.0	84.172563	-1.172563
6	17	149.5	4878	45.0	38.338373	6.661627
7	3	170.0	5245	100.0	109.560739	-9.560739
8	13	165.3	5245	65.0	62.984573	2.015427
9	11	165.1	5245	70.0	72.023750	-2.023750
10	20	149.0	5245	33.0	27.230825	5.769175
11	12	158.0	5245	63.0	65.760648	-2.760648
12	17	123.5	5245	43.0	34.687294	8.312706
13	6	170.1	5752	95.0	99.607453	-4.607453
14	6	170.1	5752	95.0	99.607453	-4.607453
15	13	151.4	5752	63.5	63.272534	0.227466
16	12	150.4	5752	67.0	67.574183	-0.574183
17	13	149.8	5752	62.0	62.885087	-0.885087
18	22	139.8	5752	31.0	19.569305	11.430695
19	11	174.5	6201	86.0	81.189822	4.810178
20	10	172.6	6201	50.5	85.273532	-34.773532
21	11	170.0	6201	64.2	80.100126	-15.900126
22	13	149.3	6618	67.0	69.005205	-2.005205
23	23	161.4	6618	28.0	26.497237	1.502763
24	23	146.0	6980	30.0	25.376962	4.623038
25	6	172.5	6980	105.0	109.038726	-4.038726
26	26	140.8	7441	22.0	13.808738	8.191262
27	5	164.2	8181	133.0	120.228162	12.771838
28	12	149.2	8886	90.0	89.870096	0.129904
29	21	146.9	8886	47.0	48.418906	-1.418906
30	23	164.5	8886	45.0	43.593219	1.406781
31	23	166.1	8886	57.0	43.980666	13.019334
32	11	171.1	9663	106.0	105.316866	0.683134
33	6	172.6	9663	152.0	128.399118	23.600882
34	14	149.5	9663	97.0	86.454915	10.545085
35	24	150.7	9663	38.0	41.307461	-3.307461
36	15	148.2	9663	87.5	81.596310	5.903690
37	20	188.3	10299	78.0	73.171294	4.828706
38	21	98.4	10299	35.0	46.857790	-11.857790
39	17	184.4	10526	83.7	87.494274	-3.794274
40	4	173.9	10526	155.0	144.021102	10.978898
41	20	135.0	10844	58.0	64.192226	-6.192226
42	19	207.1	11193	83.0	88.710595	-5.710595
43	26	137.0	11193	25.0	39.928928	-14.928928
44	9	171.7	11193	135.0	125.576360	9.423640
45	4	170.0	11614	158.0	150.917832	7.082168
46	15	161.5	12479	87.2	105.111663	-17.911663
47	18	145.4	12479	82.0	87.581562	-5.581562

	OLS1_Y_predicted	OLS1_Residuals	OLS2_Residuals	OLS2_Y_predicted
0	82.580410	-9.580410	-0.070444	82.580410

1	41.386886	3.613114	0.026567	41.386886
2	59.780041	2.219959	0.016323	59.780041
3	61.450908	-1.450908	-0.010668	61.450908
4	56.407282	4.892718	0.035976	56.407282
5	84.172563	-1.172563	-0.008622	84.172563
6	38.338373	6.661627	0.048983	38.338373
7	109.560739	-9.560739	-0.070300	109.560739
8	62.984573	2.015427	0.014819	62.984573
9	72.023750	-2.023750	-0.014881	72.023750
10	27.230825	5.769175	0.042420	27.230825
11	65.760648	-2.760648	-0.020299	65.760648
12	34.687294	8.312706	0.061123	34.687294
13	99.607453	-4.607453	-0.033878	99.607453
14	99.607453	-4.607453	-0.033878	99.607453
15	63.272534	0.227466	0.001673	63.272534
16	67.574183	-0.574183	-0.004222	67.574183
17	62.885087	-0.885087	-0.006508	62.885087
18	19.569305	11.430695	0.084049	19.569305
19	81.189822	4.810178	0.035369	81.189822
20	85.273532	-34.773532	-0.255688	85.273532
21	80.100126	-15.900126	-0.116913	80.100126
22	69.005205	-2.005205	-0.014744	69.005205
23	26.497237	1.502763	0.011050	26.497237
24	25.376962	4.623038	0.033993	25.376962
25	109.038726	-4.038726	-0.029697	109.038726
26	13.808738	8.191262	0.060230	13.808738
27	120.228162	12.771838	0.093911	120.228162
28	89.870096	0.129904	0.000955	89.870096
29	48.418906	-1.418906	-0.010433	48.418906
30	43.593219	1.406781	0.010344	43.593219
31	43.980666	13.019334	0.095730	43.980666
32	105.316866	0.683134	0.005023	105.316866
33	128.399118	23.600882	0.173536	128.399118
34	86.454915	10.545085	0.077537	86.454915
35	41.307461	-3.307461	-0.024320	41.307461
36	81.596310	5.903690	0.043409	81.596310
37	73.171294	4.828706	0.035505	73.171294
38	46.857790	-11.857790	-0.087190	46.857790
39	87.494274	-3.794274	-0.027899	87.494274
40	144.021102	10.978898	0.080727	144.021102
41	64.192226	-6.192226	-0.045531	64.192226
42	88.710595	-5.710595	-0.041990	88.710595
43	39.928928	-14.928928	-0.109772	39.928928
44	125.576360	9.423640	0.069291	125.576360
45	150.917832	7.082168	0.052075	150.917832
46	105.111663	-17.911663	-0.131703	105.111663
47	87.581562	-5.581562	-0.041041	87.581562

[123]:
```
ols2.summary()
```

[123]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.920
Model:                            OLS   Adj. R-squared:                  0.915
Method:                 Least Squares   F-statistic:                     169.7
Date:                Thu, 10 Nov 2022   Prob (F-statistic):           3.39e-24
Time:                        12:37:43   Log-Likelihood:                 59.835
No. Observations:                  48   AIC:                            -111.7
Df Residuals:                      44   BIC:                            -104.2
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4847      0.054      9.002      0.000       0.376       0.593
x1            -0.7684      0.044    -17.378      0.000      -0.858      -0.679
x2             0.1935      0.073      2.643      0.011       0.046       0.341
x3             0.4150      0.034     12.051      0.000       0.346       0.484
==============================================================================
Omnibus:                       13.373   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               19.393
Skew:                          -0.851   Prob(JB):                     6.15e-05
Kurtosis:                       5.607   Cond. No.                         11.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Exhibit 18

[124]:
```
sns.distplot(ols2_residuals, color = 'purple')
plt.title("Residual Plot for OLS with Standardization")
plt.xlabel("Residuals")
```

[124]:
```
Text(0.5, 0, 'Residuals')
```

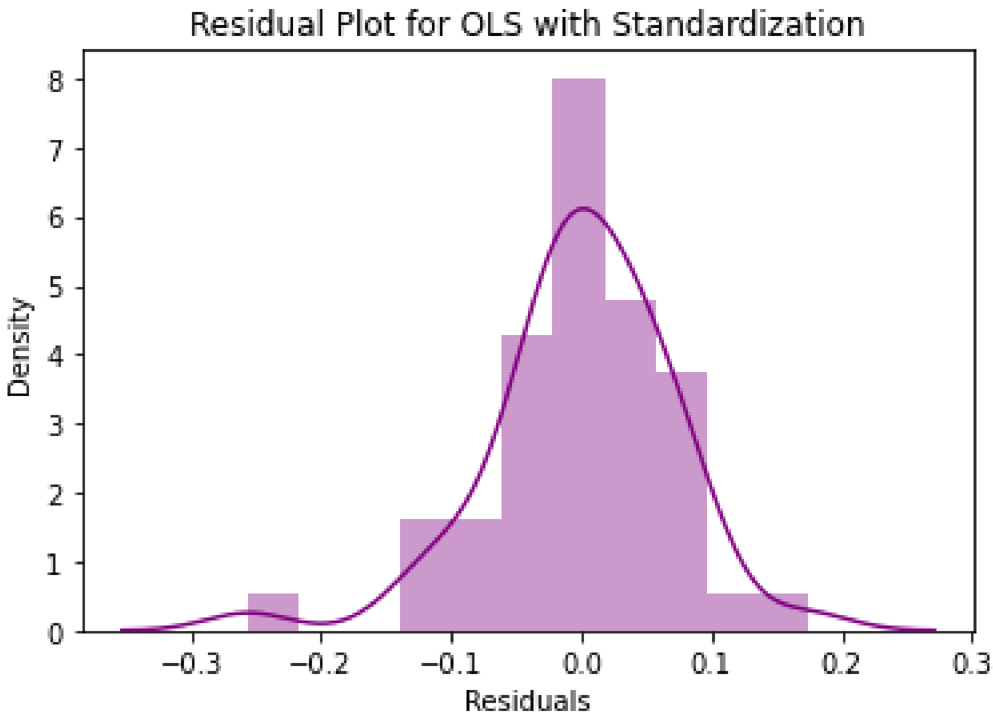
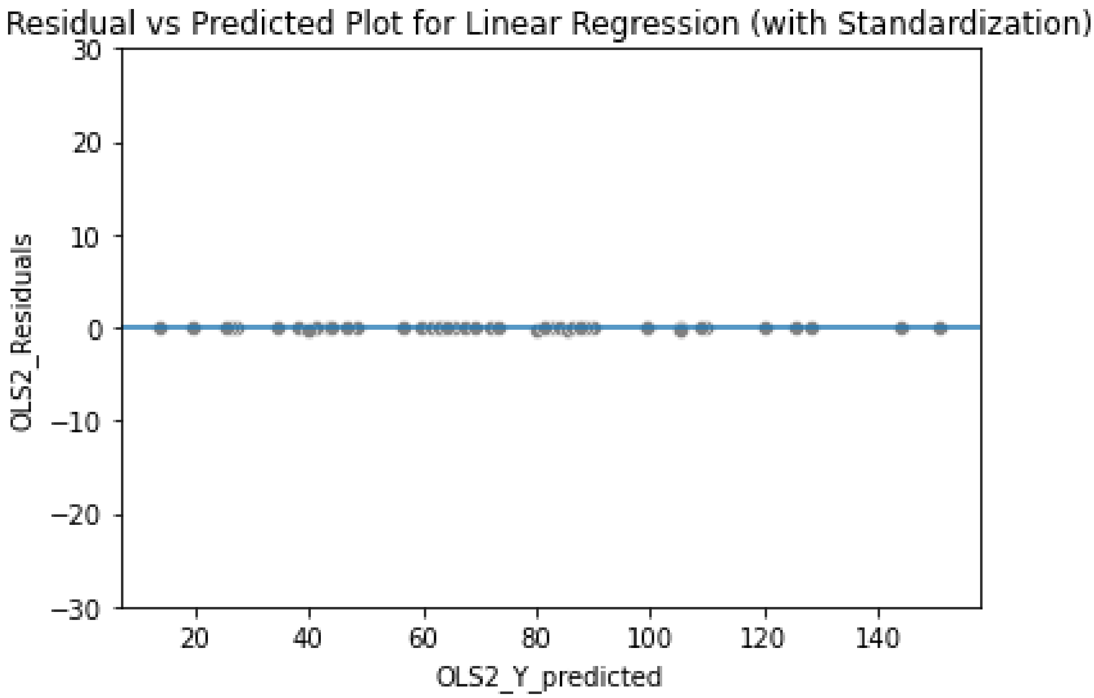


Exhibit 19

[125]:
```
sns.scatterplot(data = shipset_regression_results, x = 'OLS2_Y_predicted' , y =␣
↪'OLS2_Residuals', color = 'grey')
plt.title("Residual vs Predicted Plot for Linear Regression (with␣
↪Standardization)")
plt.axhline(y=0)
plt.ylim(-30,30)
```

[125]: (-30.0, 30.0)



[126]:
```
print(mean_absolute_error(Y, ols2_predictions))
```

```
0.050442459352507706
```

3 Synthetic Data: Created by Gretel.AI with 86% similarity rate

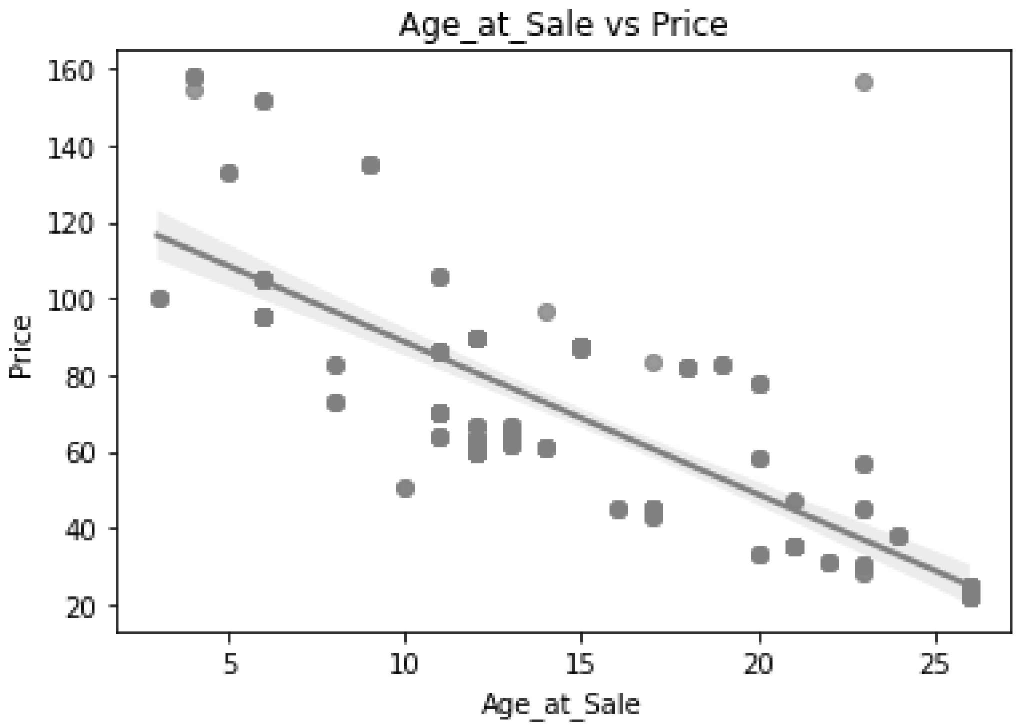
[127]:
```
synthdata = pd.read_csv('syntheticdata.csv') ; synthdata ;  synthdata.
 ↪rename(columns={"Age at Sale": "Age_at_Sale"},inplace = True)
```

[128]:
```
synthdata.drop('Vessel',axis = 1,inplace = True)
```

[129]:
```
synthdata = synthdata.loc[3000:3249]
```

[130]:
```
sns.regplot(x = synthdata['Age_at_Sale'], y = synthdata['Price'],marker = 'o',␣
 ↪color = 'grey')
plt.title('Age_at_Sale vs Price')
```

[130]:
```
Text(0.5, 1.0, 'Age_at_Sale vs Price')
```

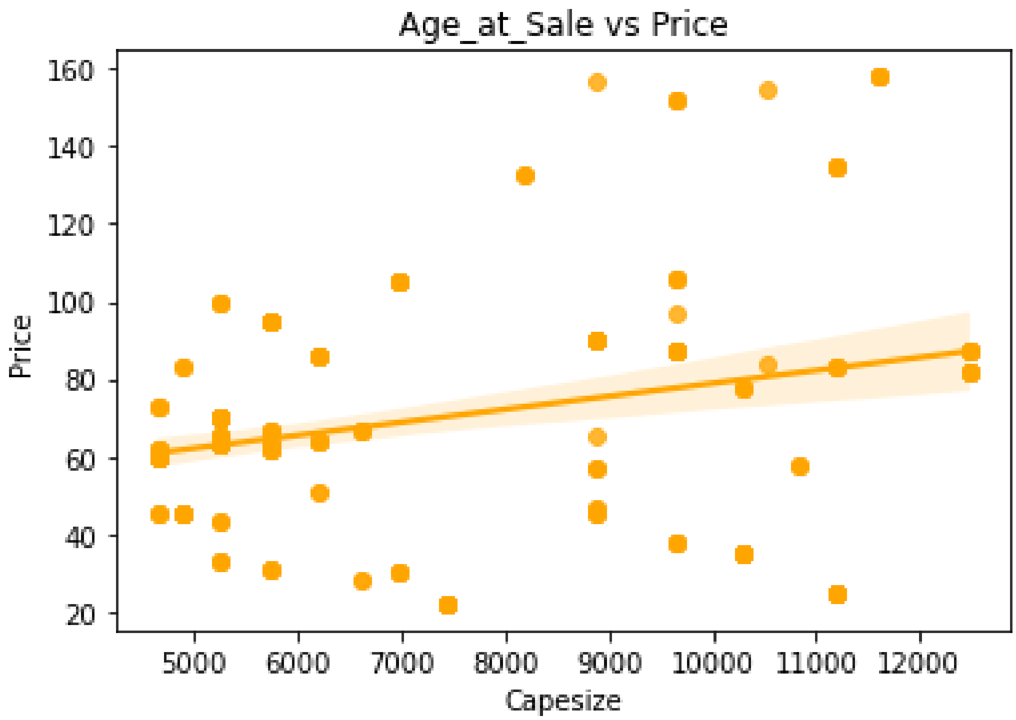


[131]:
```
sns.regplot(x = synthdata['DWT'], y = synthdata['Price'],marker = 'o', color =␣
↪'red')
plt.title('Age_at_Sale vs Price')
```

[131]: Text(0.5, 1.0, 'Age_at_Sale vs Price')

[132]:
```
sns.regplot(x = synthdata['Capesize'], y = synthdata['Price'],marker = 'o',␣
    ↪color = 'orange')
plt.title('Age_at_Sale vs Price')
```

[132]: Text(0.5, 1.0, 'Age_at_Sale vs Price')



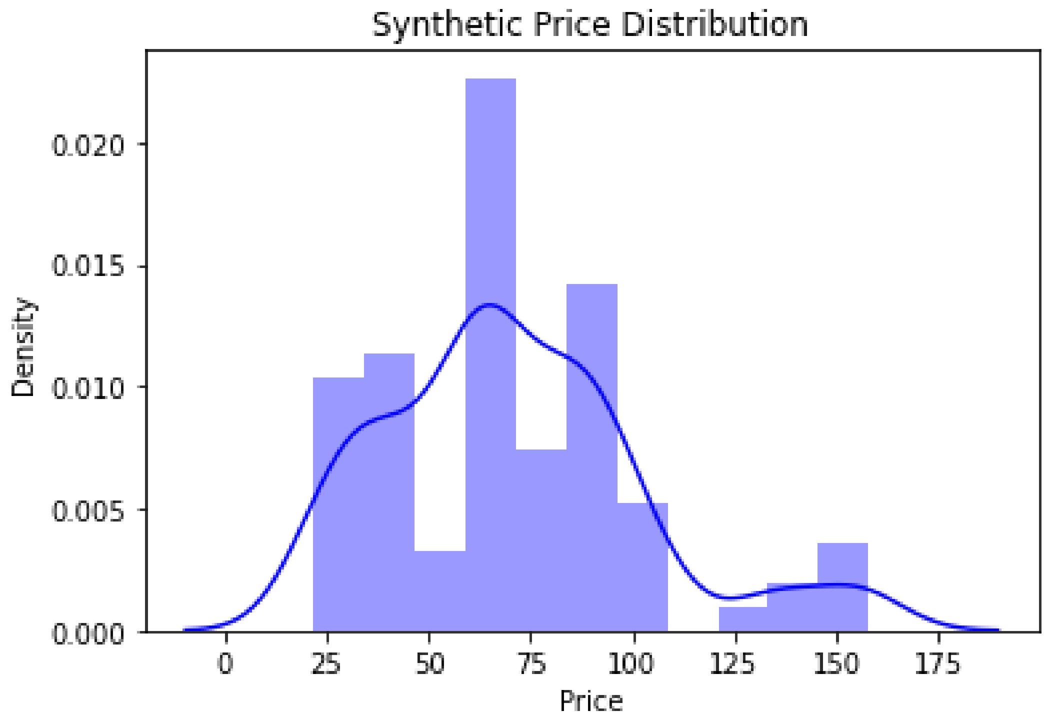
[133]:
```
synthdata.corr()
```

[133]:

	Price	YearBuilt	Age_at_Sale	DWT	Capesize	Month
Price	1.000000	0.783345	-0.764145	0.547482	0.266079	0.061637
YearBuilt	0.783345	1.000000	-0.998150	0.503380	-0.252008	-0.286505
Age_at_Sale	-0.764145	-0.998150	1.000000	-0.494307	0.293563	0.270693
DWT	0.547482	0.503380	-0.494307	1.000000	-0.130340	-0.292551
Capesize	0.266079	-0.252008	0.293563	-0.130340	1.000000	0.519398
Month	0.061637	-0.286505	0.270693	-0.292551	0.519398	1.000000

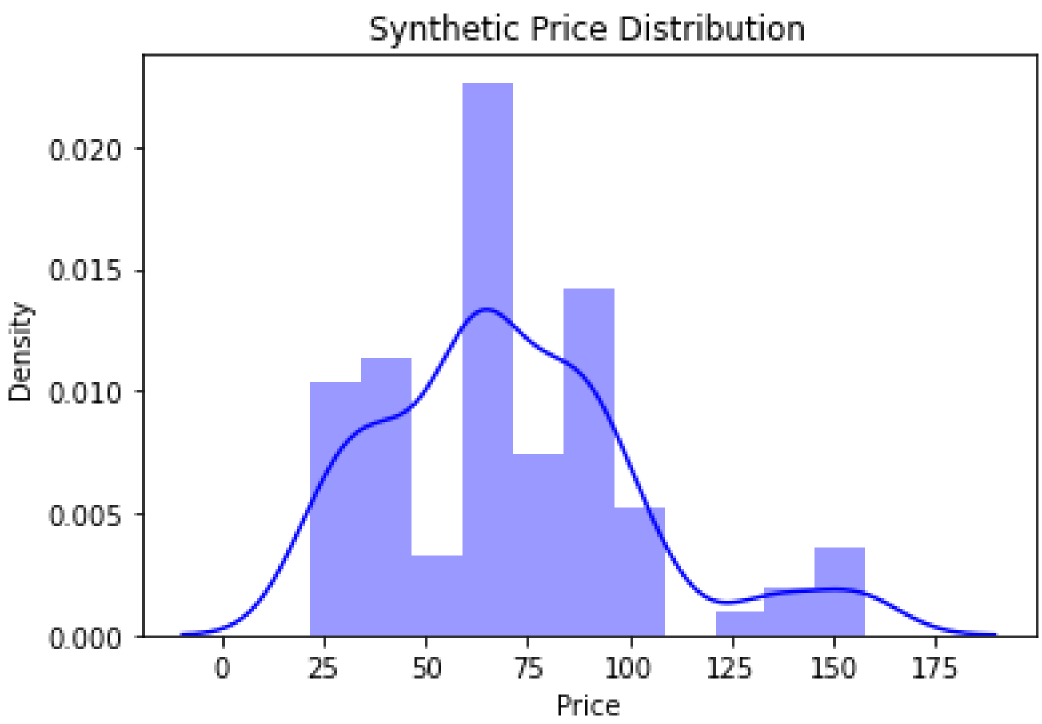
[134]:
```
sns.distplot(synthdata['Price'], color = 'blue')
plt.title("Synthetic Price Distribution")
```

[134]: Text(0.5, 1.0, 'Synthetic Price Distribution')




```
[135]: sns.distplot(synthdata['Price'], color = 'blue')
       plt.title("Synthetic Price Distribution")
```

```
[135]: Text(0.5, 1.0, 'Synthetic Price Distribution')
```



[136]:
```
fake = Faker()
```

[137]:
```
data = [
            {
              "Vessel": fake.company(),
            }
            for i in range(5000)
        ]
```

[138]:
```
fake_vessel = pd.DataFrame(data = data) ; fake_vessel ; synthdata['Vessel'] =␣
 ↪fake_vessel
```

4 Linear Regression by Treating Original Dataset as Training and Synthetic Data as Test

[139]:
```
X_train = shipset_regression.values[:,:-1] ;Y_train = shipset_regression.
 ↪values[:,-1]
```

[140]:
```
synthdata_regression = synthdata[["Age_at_Sale","DWT","Capesize","Price"]].
 ↪copy() ;
```

[141]:
```
X_test = synthdata_regression.values[:,:-1]; Y_test = synthdata_regression.
 ↪values[:,-1]
```

[142]:
```
LR_synth = LinearRegression().fit(X_train, Y_train)
```

[143]:
```
LR_synth_predicted = LR_synth.predict(X_test)
```

[144]:
```
LR_synth.score(X_test,Y_test)
```

[144]:
```
0.8766474942576405
```

[145]:
```
Y_synth_predicted = LR_synth.predict(X_test)
```

[146]:
```
LR_Synth_Residuals = Y_test - Y_synth_predicted
```

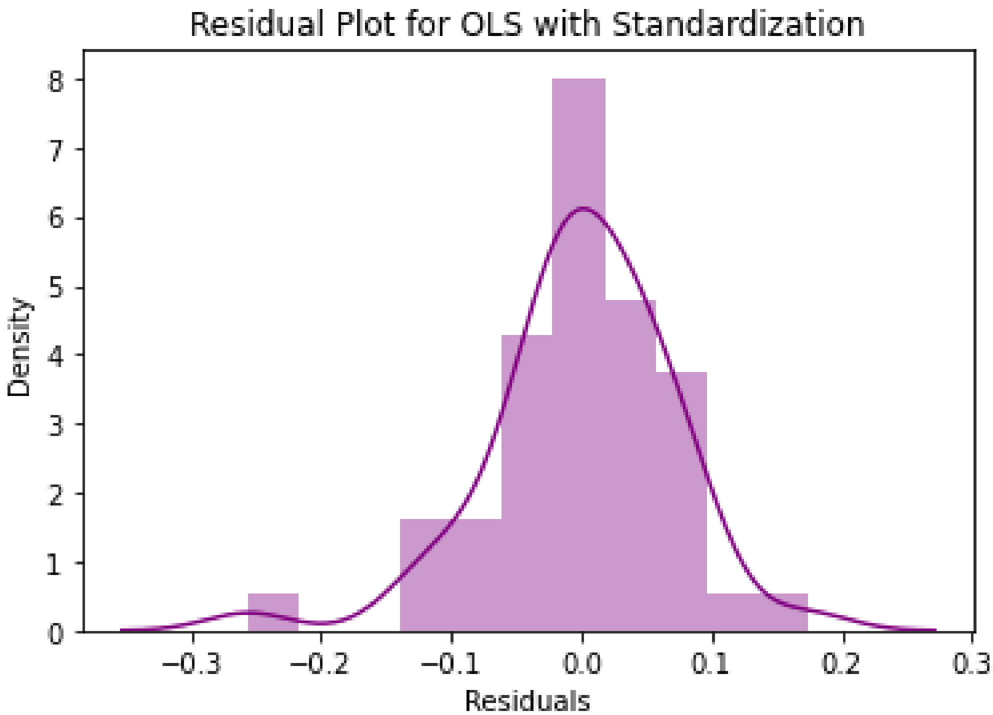
[147]:
```
#synthdata_regression['LR_Synth_Predicted Values'] = Y_synth_predicted ;
#synthdata_regression['LR_Synth_Residuals'] = LR_Synth_Residuals
```

[148]:
```
print(mean_absolute_error(Y_test, Y_synth_predicted)) ;

print(mean_squared_error(Y_test, Y_synth_predicted))
```

```
6.9132861257138725
124.20078328000722
```

[149]:
```
sns.distplot(ols2_residuals, color = 'purple')
plt.title("Residual Plot for OLS with Standardization")
plt.xlabel("Residuals")
```

[149]:
```
Text(0.5, 0, 'Residuals')
```



[150]:
```
m = LR_synth.coef_.flatten() ; b = LR_synth.intercept_.flatten() ; print("m =␣
↪{0}".format(m)) ; print("b = {0}".format(b))
```

```
m = [-4.54380392  0.24215462  0.00720692]
b = [44.22554998]
```

5 OLS by Treating Original Dataset as Training and Synthetic Data as Test

Exhibit 20

[151]:
```
synthdata.describe()
```

[151]:

	Price	YearBuilt	Age_at_Sale	DWT	Capesize	\
count	250.000000	250.000000	250.000000	250.000000	250.000000	
mean	70.704800	1992.668000	14.496000	156.716000	7567.416000	
std	31.794977	6.065057	6.094469	18.951273	2523.703905	
min	22.000000	1981.000000	3.000000	98.400000	4647.000000	
25%	45.000000	1988.000000	11.000000	149.000000	5245.000000	
50%	65.000000	1994.000000	13.000000	158.000000	6618.000000	
75%	87.500000	1996.000000	20.000000	170.100000	9663.000000	
max	158.000000	2004.000000	26.000000	207.100000	12479.000000	

```
             Month
```

```
count    250.000000
mean       5.436000
std        3.556473
min        1.000000
25%        3.000000
50%        4.000000
75%        8.750000
max       12.000000
```

[152]:
```
shipset.describe()
```

[152]:
```
           Price    YearBuilt  Age_at_Sale         DWT      Capesize  \
count   48.00000    48.000000    48.000000   48.000000     48.000000
mean    72.95625  1992.916667    14.270833  158.935417   7643.708333
std     33.89537     6.330720     6.330405   17.650984   2499.309368
min     22.00000  1981.000000     3.000000   98.400000   4647.000000
25%     46.50000  1987.750000    10.750000  149.275000   5245.000000
50%     66.00000  1994.000000    13.000000  161.450000   6799.000000
75%     88.12500  1996.250000    20.000000  170.125000   9663.000000
max    158.00000  2004.000000    26.000000  207.100000  12479.000000

           Month
count  48.000000
mean    5.312500
std     3.543987
min     1.000000
25%     3.000000
50%     4.000000
75%     8.250000
max    12.000000
```

[153]:
```
Y_synth, X_synth = dmatrices('Price ~ DWT+Age_at_Sale+Capesize',␣
 ↪data=synthdata_regression, return_type='dataframe')
```

[154]:
```
vif = pd.DataFrame()
vif['VIF'] = [variance_inflation_factor(X_synth.values, i) for i in␣
 ↪range(X_synth.shape[1])]
vif['variable'] = X_synth.columns ; vif
```

[154]:
```
          VIF     variable
0  130.130663    Intercept
1    1.323765          DWT
2    1.423994  Age_at_Sale
3    1.094652     Capesize
```

[155]:
```
X_test = sm.add_constant(X_test) ; X_train = sm.add_constant(X_train)
```

[156]:
```
ols_synth = sm.OLS(Y_train, X_train).fit()
```

[157]:
```
predictions_synth = ols_synth.predict(X_test)
```

[158]:
```
#synthdata_regression['OLS_Synth_Predicted Values'] = predictions_synth
#synthdata_regression['OLS_Synth_Residuals']= abs(synthdata_regression['Price']␣
↪- synthdata_regression['OLS_Synth_Predicted Values'])
```

[159]:
```
print(mean_absolute_error(Y_test, predictions_synth))
```

```
6.913286125713921
```

[160]:
```
print(mean_squared_error(Y_test, predictions_synth))
```

```
124.20078328001003
```

[161]:
```
ols_synth.summary()
```

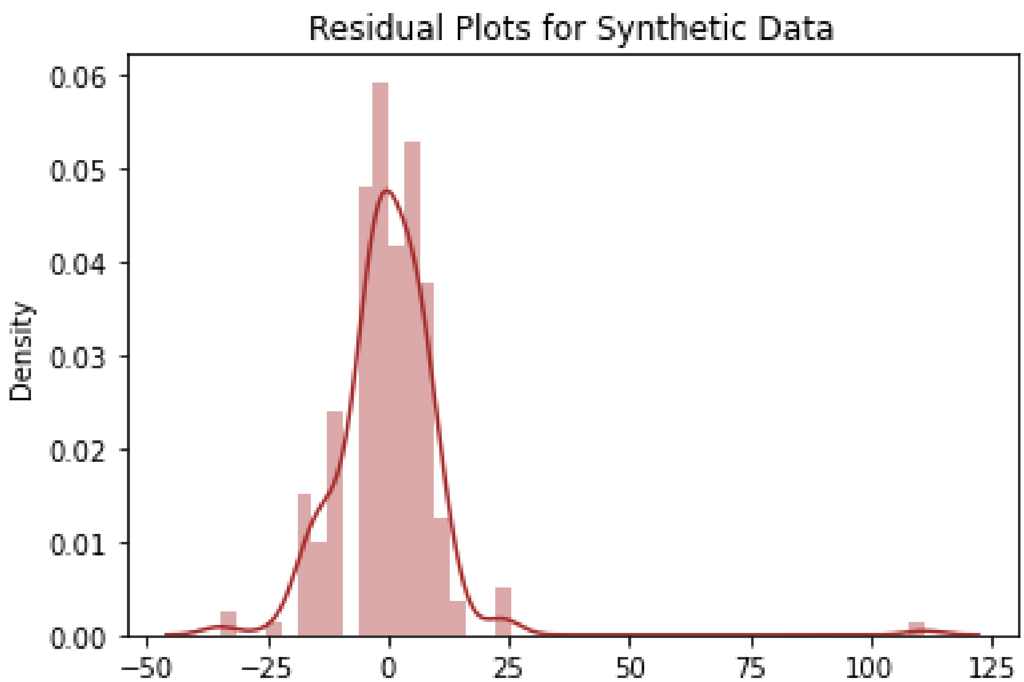
[161]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.920
Model:                            OLS   Adj. R-squared:                  0.915
Method:                 Least Squares   F-statistic:                     169.7
Date:                Thu, 10 Nov 2022   Prob (F-statistic):           3.39e-24
Time:                        12:37:44   Log-Likelihood:                -175.97
No. Observations:                  48   AIC:                             359.9
Df Residuals:                      44   BIC:                             367.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         44.2255     16.383      2.699      0.010      11.207      77.244
x1            -4.5438      0.261    -17.378      0.000      -5.071      -4.017
x2             0.2422      0.092      2.643      0.011       0.058       0.427
x3             0.0072      0.001     12.051      0.000       0.006       0.008
==============================================================================
Omnibus:                       13.373   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               19.393
Skew:                          -0.851   Prob(JB):                     6.15e-05
Kurtosis:                       5.607   Cond. No.                     9.23e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
[2] The condition number is large, 9.23e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Exhibit 21

```
[162]: sns.distplot(LR_Synth_Residuals,color = 'brown')
       plt.title("Residual Plots for Synthetic Data")
```

```
[162]: Text(0.5, 1.0, 'Residual Plots for Synthetic Data')
```




```
[164]: sns.scatterplot(x = predictions_synth , y = LR_Synth_Residuals, color = 'grey')
       plt.title("Residual vs Predicted Plot for Linear Regression on Synthetic Data")
       plt.axhline(y=0)
       plt.ylim(-75,75)
```

```
[164]: (-75.0, 75.0)
```

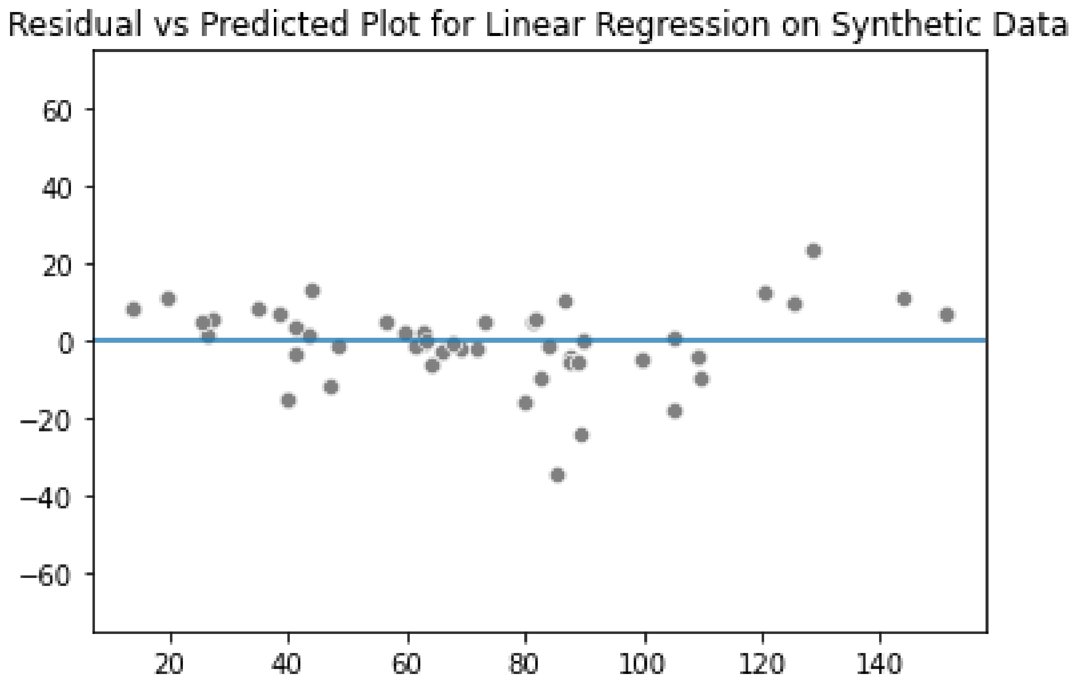


Exhibit 22 #

Thank You

```
[ ]:
```