Title: Linear Regression vs Polynomial Regression 📈

Goal 🎯 : Find which model works best without overfitting using Advertising dataset.

Lets start with importing the libraries and loading the dataset

In [21]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [22]:
```
advertising_df = pd.read_csv('Advertising.csv')
df = advertising_df.copy()
df.head()
```

Out[22]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Lets check if the data is clean before we start the comparison.

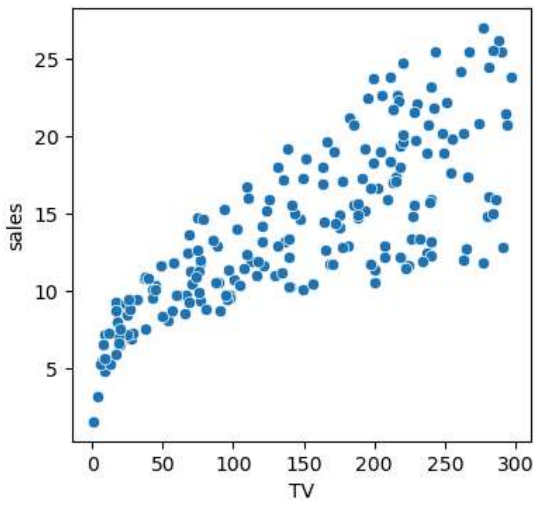
In [23]:
```
df.info()
```

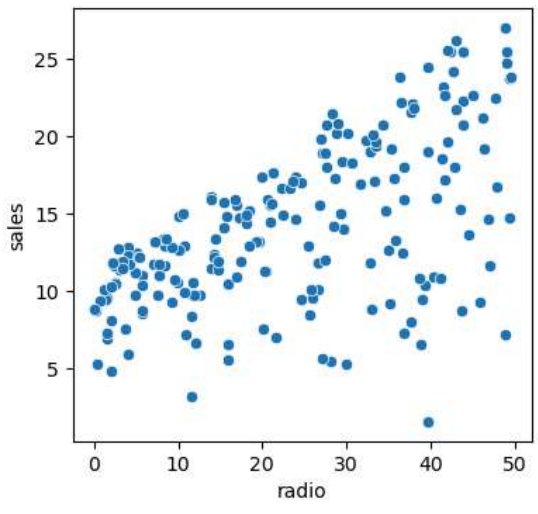
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

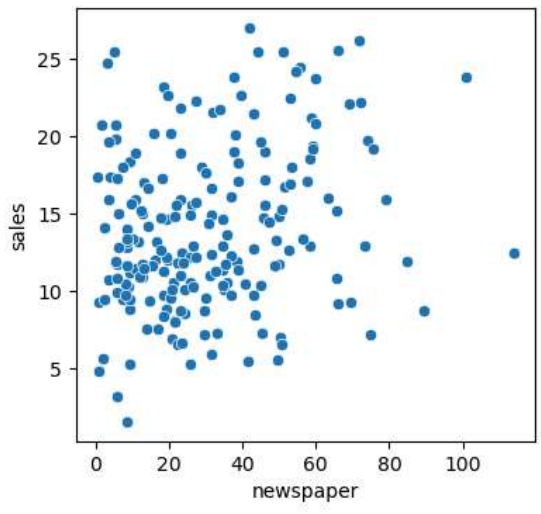
In [24]:
```
df.isna().sum()
```

Out[24]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [25]:
```
plt.figure(figsize=(16,4))
plt.subplot(1,3,1)
sns.scatterplot(data=df,x='TV',y='sales')

plt.subplot(1,3,2)
sns.scatterplot(data=df,x='radio',y='sales')

plt.subplot(1,3,3)
sns.scatterplot(data=df,x='newspaper',y='sales')

plt.subplots_adjust(wspace=0.4,hspace=0.4)
```







In [26]:
```
df.describe().T
```

Out[26]:

	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
sales	200.0	14.0225	5.217457	1.6	10.375	12.90	17.400	27.0

The data is cleaned where there is no null-values nor zero values that doesnt make any sense.

It is time to train the models.

In [27]:
```
X = df.drop('sales',axis=1)
y = df['sales']
```

In [28]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

First, lets start with Linear Regression.

In [29]:
```
from sklearn.linear_model import LinearRegression
lr_model = LinearRegression()
lr_model.fit(X_train,y_train)
```

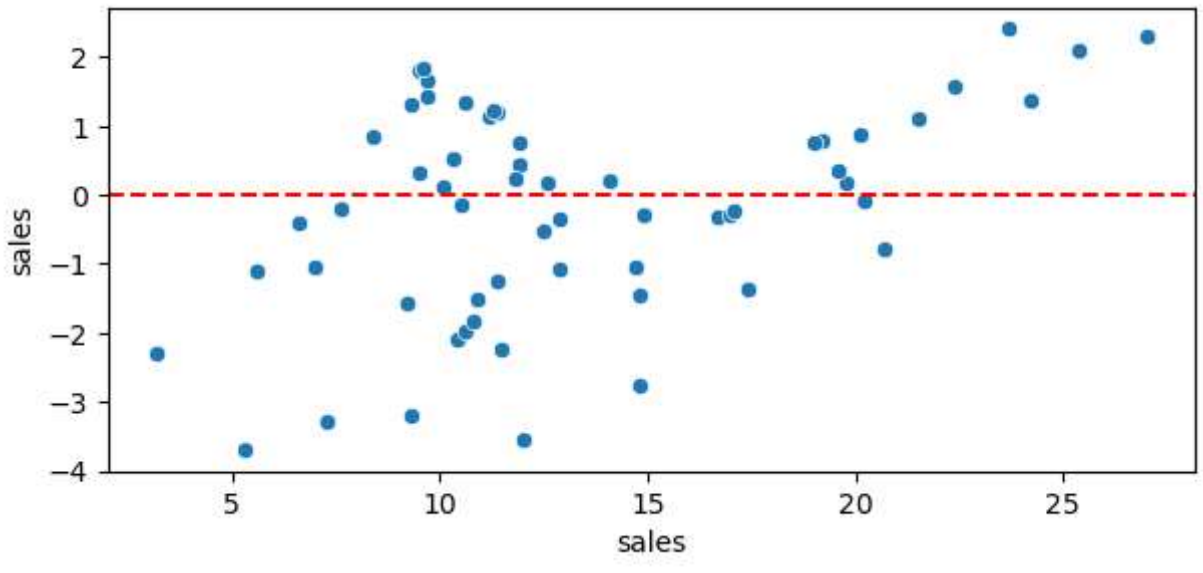
Out[29]:
```
LinearRegression
LinearRegression()
```

In [30]:
```
test_prediction = lr_model.predict(X_test)
```

I want to visualize using residual plot to make sure that Linear Regression can be used in this dataset.

In [31]:
```
test_residuals = y_test - test_prediction
plt.figure(figsize=(7,3))
sns.scatterplot(x=y_test,y=test_residuals)
plt.axhline(y=0,color='red',ls='--')
```

Out[31]:
```
<matplotlib.lines.Line2D at 0x255b9a9f4d0>
```



Now it is time for Polynomial Regression.

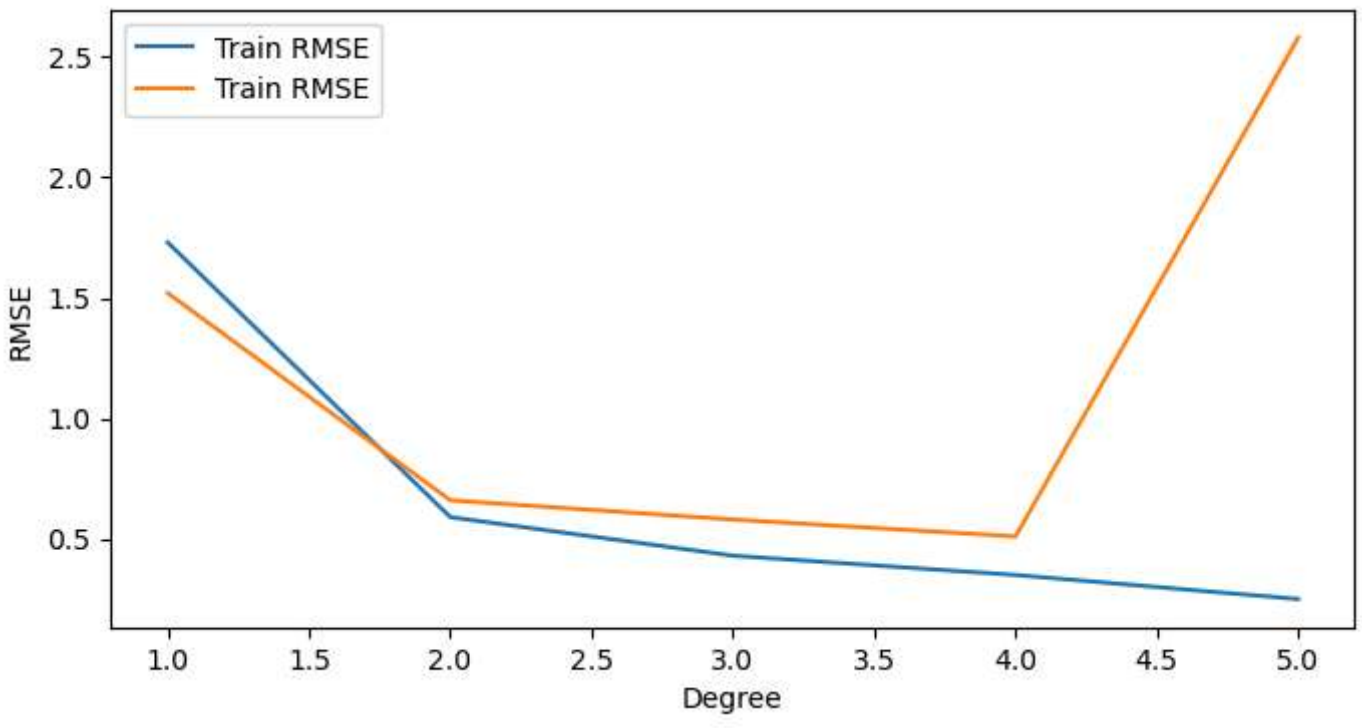
In [32]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

Im going to use for loop to check which degree is the best to use.

In [33]:
```
from sklearn.metrics import r2_score, root_mean_squared_error
train_rmse_errors = []
test_rmse_errore = []
train_r_errors = []
test_r_errors = []

for i in range(1,10):
    poly_converter = PolynomialFeatures(degree=i,include_bias=False)
    poly_features = poly_converter.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.3, random_state=101)
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    pr_model = LinearRegression()
    pr_model.fit(X_train,y_train)
    train_pred = pr_model.predict(X_train)
    test_pred = pr_model.predict(X_test)
    train_rmse = root_mean_squared_error(y_train,train_pred).round(2)
    test_rmse = root_mean_squared_error(y_test,test_pred).round(2)
    train_r = round(r2_score(y_train,train_pred),2)
    test_r = round(r2_score(y_test,test_pred),2)
    train_rmse_errors.append(train_rmse)
    test_rmse_errore.append(test_rmse)
    train_r_errors.append(train_r)
    test_r_errors.append(test_r)
    print(f'Train RMSE for degree {i}: {train_rmse}')
    print(f'Test RMSE for degree {i}: {test_rmse}')
    print(f'Train R^2 for degree {i}: {train_r}')
    print(f'Test R^2 for degree {i}: {test_r}')
    print()
```

```
Train RMSE for degree 1: 1.73
Test RMSE for degree 1: 1.52
Train R^2 for degree 1: 0.89
Test R^2 for degree 1: 0.92

Train RMSE for degree 2: 0.59
Test RMSE for degree 2: 0.66
Train R^2 for degree 2: 0.99
Test R^2 for degree 2: 0.98

Train RMSE for degree 3: 0.43
Test RMSE for degree 3: 0.58
Train R^2 for degree 3: 0.99
Test R^2 for degree 3: 0.99

Train RMSE for degree 4: 0.35
Test RMSE for degree 4: 0.51
Train R^2 for degree 4: 1.0
Test R^2 for degree 4: 0.99

Train RMSE for degree 5: 0.25
Test RMSE for degree 5: 2.58
Train R^2 for degree 5: 1.0
Test R^2 for degree 5: 0.76

Train RMSE for degree 6: 0.19
Test RMSE for degree 6: 7.23
Train R^2 for degree 6: 1.0
Test R^2 for degree 6: -0.85

Train RMSE for degree 7: 0.11
Test RMSE for degree 7: 113.15
Train R^2 for degree 7: 1.0
Test R^2 for degree 7: -452.45

Train RMSE for degree 8: 0.0
Test RMSE for degree 8: 1396.85
Train R^2 for degree 8: 1.0
Test R^2 for degree 8: -69110.83

Train RMSE for degree 9: 0.0
Test RMSE for degree 9: 2342.54
Train R^2 for degree 9: 1.0
Test R^2 for degree 9: -194368.78
```

In [34]:
```
plt.figure(figsize=(8,4))
plt.plot(range(1,6),train_rmse_errors[:5],label='Train RMSE')
plt.plot(range(1,6),test_rmse_errore[:5],label='Train RMSE')
plt.xlabel('Degree')
plt.ylabel('RMSE')
plt.legend()
plt.show()
```



From the results. I decided to use the 3rd degree since it gave us the best results while not being overfit.

In [35]:
```
final_poly_converter = PolynomialFeatures(degree=3,include_bias=False)
poly_features = final_poly_converter.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.3, random_state=101)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
final_pr_model = LinearRegression()
```

```
final_pr_model.fit(X_train,y_train)
final_pr_pred = final_pr_model.predict(X_test)
```

If we are still not sure if we have overfitting, lets use regularizations to prevent from overfitting. I will strt with Ridge Regression with Cross Validation to give us the best alpha to use.

In [36]:
```
from sklearn.linear_model import RidgeCV
# help(RidgeCV)
ridge_cv_model = RidgeCV(alphas=(0.1,1.0,10.0),cv=10,)
ridge_cv_model.fit(X_train,y_train)
ridge_cv_pred_values = ridge_cv_model.predict(X_test)
```

In [37]:
```
poly_converter.get_feature_names_out()[0:19]
```

Out[37]:
```
array(['TV', 'radio', 'newspaper', 'TV^2', 'TV radio', 'TV newspaper',
       'radio^2', 'radio newspaper', 'newspaper^2', 'TV^3', 'TV^2 radio',
       'TV^2 newspaper', 'TV radio^2', 'TV radio newspaper',
       'TV newspaper^2', 'radio^3', 'radio^2 newspaper',
       'radio newspaper^2', 'newspaper^3'], dtype=object)
```

In [38]:
```
ridge_cv_model.coef_
```

Out[38]:
```
array([ 5.40769392,  0.5885865 ,  0.40390395, -6.18263924,  4.59607939,
       -1.18789654, -1.15200458,  0.57837796, -0.1261586 ,  2.5569777 ,
       -1.38900471,  0.86059434,  0.72219553, -0.26129256,  0.17870787,
        0.44353612, -0.21362436, -0.04622473, -0.06441449])
```

The other regularization to use is the Lasso Regression with Cross Validation.

In [39]:
```
from sklearn.linear_model import LassoCV
# help(LassoCV)
lasso_cv_model = LassoCV(cv=10,eps=0.001,n_alphas=100,max_iter=1000000)
lasso_cv_model.fit(X_train,y_train)
lasso_pred = lasso_cv_model.predict(X_test)
```

In [40]:
```
poly_converter.get_feature_names_out()[0:19]
```

Out[40]:
```
array(['TV', 'radio', 'newspaper', 'TV^2', 'TV radio', 'TV newspaper',
       'radio^2', 'radio newspaper', 'newspaper^2', 'TV^3', 'TV^2 radio',
       'TV^2 newspaper', 'TV radio^2', 'TV radio newspaper',
       'TV newspaper^2', 'radio^3', 'radio^2 newspaper',
       'radio newspaper^2', 'newspaper^3'], dtype=object)
```

In [41]:
```
lasso_cv_model.coef_
```

Out[41]:
```
array([ 4.86023329,  0.12544598,  0.20746872, -4.99250395,  4.38026519,
       -0.22977201, -0.        ,  0.07267717, -0.        ,  1.77780246,
       -0.69614918, -0.        ,  0.12044132, -0.        , -0.        ,
       -0.        ,  0.        ,  0.        , -0.        ])
```

Last one is Elastic-Net, which is mix between Ridge and Lasso Regressions.

In [42]:
```
from sklearn.linear_model import ElasticNetCV
elastic_cv_model = ElasticNetCV(l1_ratio=[.1,.2,.3,.4,.5,.6,.7,.8,.9,.95,1],eps=0.001,cv=10,n_alphas=100,max_iter=1000000)
elastic_cv_model.fit(X_train,y_train)
elastic_pred = elastic_cv_model.predict(X_test)
```

In [43]:
```
elastic_cv_model.l1_ratio_
```

Out[43]:
```
1.0
```

In [44]:
```
poly_converter.get_feature_names_out()[0:19]
```

Out[44]:
```
array(['TV', 'radio', 'newspaper', 'TV^2', 'TV radio', 'TV newspaper',
       'radio^2', 'radio newspaper', 'newspaper^2', 'TV^3', 'TV^2 radio',
       'TV^2 newspaper', 'TV radio^2', 'TV radio newspaper',
       'TV newspaper^2', 'radio^3', 'radio^2 newspaper',
       'radio newspaper^2', 'newspaper^3'], dtype=object)
```

In [45]:
```
elastic_cv_model.coef_
```

Out[45]:
```
array([ 4.86023329,  0.12544598,  0.20746872, -4.99250395,  4.38026519,
       -0.22977201, -0.        ,  0.07267717, -0.        ,  1.77780246,
       -0.69614918, -0.        ,  0.12044132, -0.        , -0.        ,
       -0.        ,  0.        ,  0.        , -0.        ])
```

Lastly, it is time to evaluate the models.

In [46]:
```
print('Linear Regression:')
print(f'RMSE: {root_mean_squared_error(y_test,test_prediction).round(2)}')
print(f'R^2: {round(r2_score(y_test,test_prediction),2)}')
print()
print('Polynomial Regression, Degree 3:')
print(f'RMSE: {root_mean_squared_error(y_test,final_pr_pred).round(2)}')
print(f'R^2: {round(r2_score(y_test,final_pr_pred),2)}')
print()
print('Ridge CV Regression:')
print(f'RMSE: {root_mean_squared_error(y_test,ridge_cv_pred_values).round(2)}')
print(f'R^2: {round(r2_score(y_test,ridge_cv_pred_values),2)}')
print()
print('Lasso CV Regression:')
print(f'RMSE: {root_mean_squared_error(y_test,lasso_pred).round(2)}')
print(f'R^2: {round(r2_score(y_test,lasso_pred),2)}')
print()
print('Elastic-Net CV Regression:')
print(f'RMSE: {root_mean_squared_error(y_test,elastic_pred).round(2)}')
print(f'R^2: {round(r2_score(y_test,elastic_pred),2)}')
```

```
Linear Regression:
RMSE: 1.52
R^2: 0.92

Polynomial Regression, Degree 3:
RMSE: 0.58
R^2: 0.99

Ridge CV Regression:
RMSE: 0.62
R^2: 0.99

Lasso CV Regression:
RMSE: 0.61
R^2: 0.99

Elastic-Net CV Regression:
RMSE: 0.61
R^2: 0.99
```

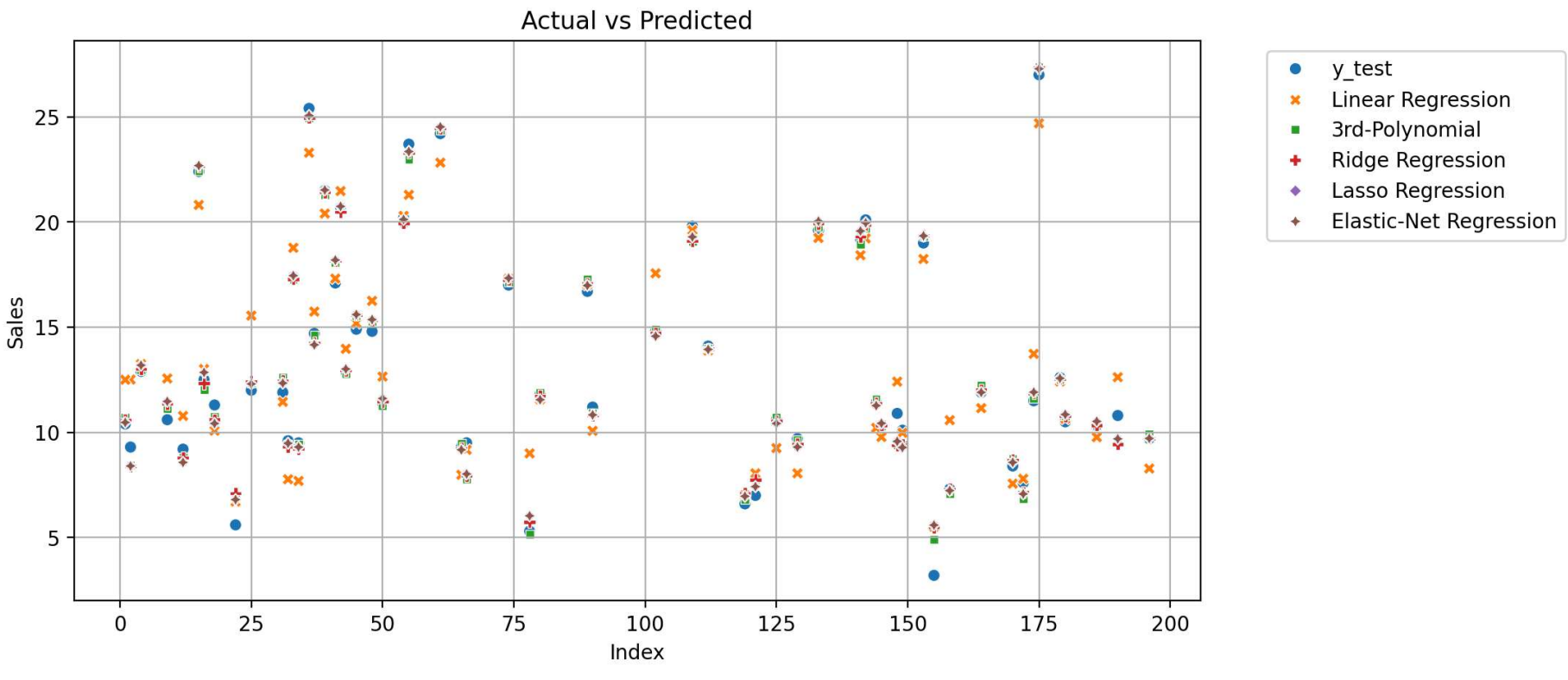
In [47]:
```
model_predictions_check = pd.DataFrame({'y_test':y_test,
                                        'Linear Regression':test_prediction.round(2),
                                        '3rd-Polynomial':final_pr_pred.round(2),
                                        'Ridge Regression':ridge_cv_pred_values.round(2),
                                        'Lasso Regression':lasso_pred.round(2),
                                        'Elastic-Net Regression':elastic_pred.round(2)})
model_predictions_check
```

Out[47]:

	y_test	Linear Regression	3rd-Polynomial	Ridge Regression	Lasso Regression	Elastic-Net Regression
37	14.7	15.74	14.62	14.25	14.16	14.16
109	19.8	19.61	19.06	19.10	19.29	19.29
31	11.9	11.45	12.61	12.45	12.34	12.34
89	16.7	17.01	17.29	17.07	16.98	16.98
66	9.5	9.17	7.79	7.94	8.02	8.02
119	6.6	7.01	6.82	7.10	6.96	6.96
54	20.2	20.29	19.96	19.95	20.12	20.12
74	17.0	17.30	17.13	17.25	17.33	17.33
145	10.3	9.78	10.40	10.34	10.43	10.43
142	20.1	19.22	19.71	19.86	19.96	19.96
148	10.9	12.41	9.41	9.38	9.57	9.57
112	14.1	13.89	13.99	13.96	13.94	13.94
174	11.5	13.73	11.62	11.85	11.91	11.91
55	23.7	21.29	22.98	23.29	23.35	23.35
141	19.2	18.42	18.95	19.30	19.57	19.57
149	10.1	9.98	9.41	9.42	9.28	9.28
25	12.0	15.55	12.33	12.42	12.30	12.30
34	9.5	7.69	9.43	9.22	9.30	9.30
170	8.4	7.56	8.75	8.70	8.57	8.57
39	21.5	20.40	21.28	21.42	21.52	21.52
172	7.6	7.79	6.83	7.15	7.07	7.07
153	19.0	18.24	19.30	19.36	19.34	19.34
175	27.0	24.69	27.39	27.33	27.28	27.28
61	24.2	22.82	24.37	24.45	24.52	24.52
65	9.3	7.98	9.48	9.24	9.17	9.17
50	11.4	12.65	11.25	11.46	11.61	11.61
42	20.7	21.47	20.73	20.47	20.74	20.74
129	9.7	8.05	9.64	9.46	9.30	9.30
179	12.6	12.42	12.52	12.55	12.57	12.57
2	9.3	12.51	8.43	8.39	8.40	8.40
12	9.2	10.78	8.82	8.79	8.57	8.57
133	19.6	19.24	19.76	19.92	20.04	20.04
90	11.2	10.07	10.92	10.81	10.83	10.83
22	5.6	6.71	6.97	7.11	6.80	6.80
41	17.1	17.31	18.08	18.16	18.19	18.19
32	9.6	7.77	9.45	9.31	9.48	9.48
125	10.6	9.25	10.72	10.54	10.42	10.42
196	9.7	8.28	9.94	9.71	9.71	9.71
158	7.3	10.58	7.08	7.31	7.23	7.23
180	10.5	10.64	10.77	10.77	10.85	10.85
16	12.5	13.01	12.03	12.33	12.85	12.85
186	10.3	9.77	10.34	10.34	10.52	10.52
144	11.4	10.21	11.59	11.39	11.27	11.27
121	7.0	8.05	7.65	7.75	7.42	7.42
80	11.8	11.57	11.89	11.73	11.55	11.55
18	11.3	10.08	10.75	10.61	10.42	10.42
78	5.3	9.00	5.15	5.74	6.03	6.03
48	14.8	16.25	15.28	15.33	15.36	15.36

	y_test	Linear Regression	3rd-Polynomial	Ridge Regression	Lasso Regression	Elastic-Net Regression
4	12.9	13.24	12.96	13.01	13.19	13.19
15	22.4	20.81	22.43	22.65	22.68	22.68
1	10.4	12.50	10.72	10.60	10.47	10.47
43	12.9	13.97	12.78	12.95	13.01	13.01
102	14.8	17.56	14.90	14.72	14.57	14.57
164	11.9	11.15	12.25	12.02	11.90	11.90
9	10.6	12.56	11.13	11.36	11.47	11.47
155	3.2	5.51	4.92	5.48	5.59	5.59
36	25.4	23.29	24.95	24.96	25.06	25.06
190	10.8	12.62	9.50	9.43	9.69	9.69
33	17.4	18.77	17.29	17.28	17.45	17.45
45	14.9	15.19	15.54	15.59	15.60	15.60

In [48]:
```
plt.figure(figsize=(10,5),dpi=200)
sns.scatterplot(data=model_predictions_check)
plt.title('Actual vs Predicted')
plt.xlabel('Index')
plt.ylabel('Sales')
plt.legend(bbox_to_anchor=(1.05,1))
plt.grid(True)
plt.show()
```



From the results, we can understand that Polynomial Regression performed better, and there wasnt overfitting since the results of Polyonial Regression and Regularizations are so close to each other.

In [ ]: