

Benjino Chery

MTH 4330

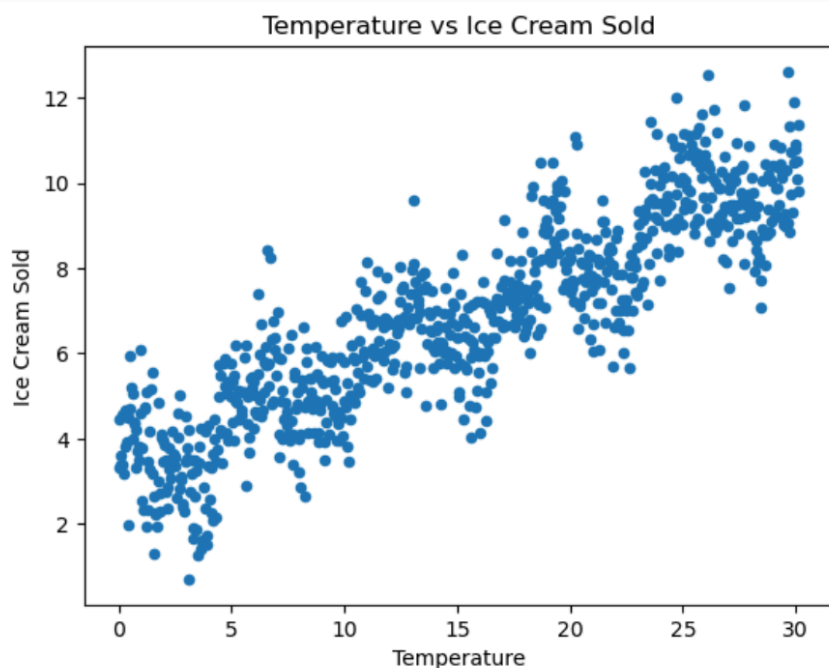
## Predictive Regression Model for Ice Cream Sales Based on Temperature Variations

```
df_tr['X'] = df_tr['X'] + 0.000001
df_tr
```

	x	y
0	0.000002	4.456329
1	0.037739	3.301815
2	0.075476	3.595955
3	0.113213	3.401495
4	0.150949	4.571147
...	...	...
795	30.000797	10.928634
796	30.038534	10.106108
797	30.076270	10.521322
798	30.114007	9.814676
799	30.151744	11.364662

	x	y
count	800.000000	800.000000
mean	15.075873	6.785090
std	8.720397	2.352965
min	0.000002	0.694817
25%	7.537937	5.008300
50%	15.075873	6.764510
75%	22.613808	8.689887
max	30.151744	12.621870

The table provides a statistical summary of 800 observations related to temperature (X) and the number of ice creams sold (y). The temperatures range from 0.000001 °C to 30.15 °C . For the number of ice cream sold, the average is approximately 6.79 units with a standard deviation of 2.35 units.



The graph provides insights into the linear regression model's assumptions. The scatter plot of temperatures versus ice cream sales shows a positive linear relationship, suggesting that as temperature increases, ice cream sales tend to rise.

## 2) Simple Linear Regression function to fit the training data.

```
reg.fit(df_tr[['X']],df_tr['y'])
```

```
LinearRegression
```

```
reg.coef_
```

```
array([0.23839763])
```

```
reg.intercept_
```

```
3.1910373234577953
```

(a) The coefficient estimates of the linear regression model are  $\beta_0 \simeq 3.1910$  and  $\beta_1 \simeq 0.2383$

(b) Since the  $\beta_1 \neq 0$ , this confirms the linear trend that can be seen in the previous scatterplot.

```
# B1 t-statistic
t_stat = reg.coef_[0]/std_B1
t_stat
```

```
53.28879296911331
```

```
pvalue = 2*(1-stats.t.cdf(t_stat,df=n-2))
pvalue
```

```
0.0
```

The t-statistic of  $\beta_1$  is 53.2887 and its p-value = 0, it can be inferred that there is an association between temperature and the number of ice cream sold.

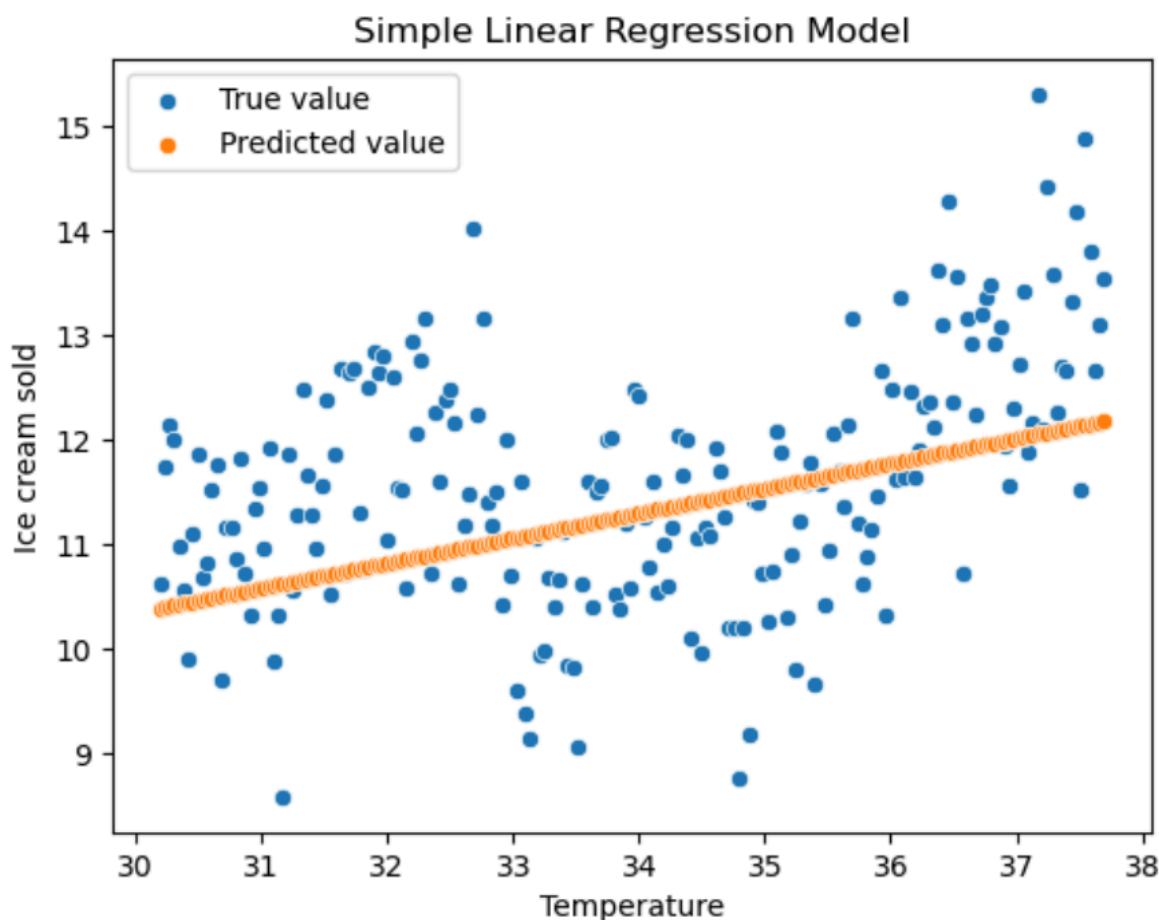
(c) The t-statistic of  $\beta_0$  is 40.9617 and its p-value is also 0 so it is significant. That is, ice creams are sold even if the temperature is low.

```
# B0 t-statistic
t_stat = reg.intercept_/std_B0
t_stat
```

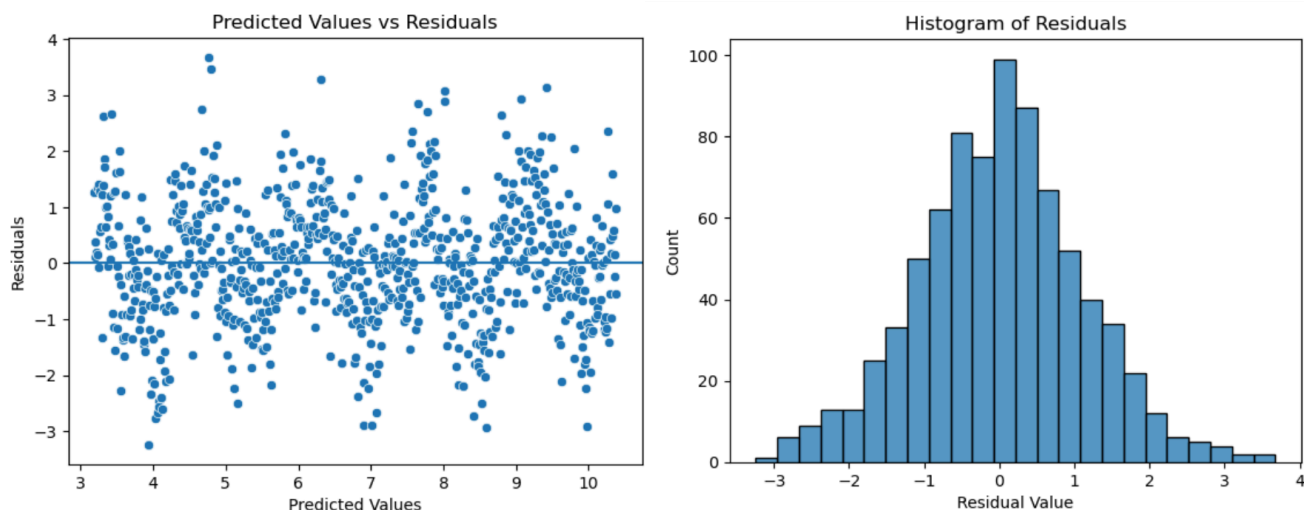
40.961717937070304

```
pvalue = 2*(1-stats.t.cdf(t_stat,df=n-2))
pvalue
```

0.0



The simple linear regression model's predictions, represented by the orange line, show a clear linear trend that increases with temperature, but fails to capture the high variability of the actual ice cream sales represented by the blue points. This suggests a more complex model that can capture nonlinear trends might improve the prediction performance.



The residuals plot displays a random scatter around zero, with no clear pattern, indicating that the assumption of constant variance is likely met. The histogram of residuals shows a roughly normal distribution, supporting the normality assumption of the errors.

### 3) Improvement of the model by adding non-linear features.

To select the best feature set for the model, both Sequential Feature Selector (**SFS**) and Recursive Feature Elimination with Cross-Validation (**RFECV**) were used, with each method evaluated based on *Adjusted R<sup>2</sup>*. SFS adds (forward selection) or removes (backward selection) features to form a feature subset in a greedy fashion. At each stage, it chooses the best feature to add or remove based on the cross-validation performance. RFECV, in contrast, recursively reduces the feature set by training the estimator and removing the least important features in each iteration. The SFS process selected features:  $X$ ,  $\cos(x)$ ,  $\sin(3x)$ ,  $\cos(2x)$ , while RFECV selected only 3:  $X$ ,  $\cos(x)$ ,  $\sin(3x)$ . In terms of performance, RFECV slightly outperformed SFS on the test set achieving an *Adjusted R<sup>2</sup>* of 0.4157 compared to SFS 0.4098. Given its performance and fewer selected features, RFECV's feature set may be preferable for better generalization and simplicity.

## Linear Regression Model based on SFS variable selection.

```
print(f"Selected features: {features}")
```

```
Selected features: ['X', 'cos(x)', 'sin(3x)', 'cos(2x)']
```

```
print(model.coef_)
```

```
[[ 0.24375646  0.80560674  0.37283376 -0.05764484]]
```

```
adjusted_r2_train = adjusted_r2_scorer(model, X_train[features], y_train)
```

```
adjusted_r2_test = adjusted_r2_scorer(model, X_test[features], y_test)
```

```
print(f"sfs Adjusted R² on the training set: {adjusted_r2_train}")
```

```
print(f"sfs Adjusted R² on the test set: {adjusted_r2_test}")
```

```
sfs Adjusted R² on the training set: 0.8525671804523196
```

```
sfs Adjusted R² on the test set: 0.4098169921744048
```

## Linear Regression Model based on RFECV variable selection.

```
print(f"Selected features: {feat}")
```

```
Selected features: ['X', 'cos(x)', 'sin(3x)']
```

```
print(model.coef_)
```

```
[[0.24390028  0.80723528  0.37146658]]
```

```
adjusted_r2_train = adjusted_r2_scorer(model, X_train[feat], y_train)
```

```
adjusted_r2_test = adjusted_r2_scorer(model, X_test[feat], y_test)
```

```
print(f"rfecv Adjusted R² on the training set: {adjusted_r2_train}")
```

```
print(f"rfecv Adjusted R² on the test set: {adjusted_r2_test}")
```

```
rfecv Adjusted R² on the training set: 0.8524488858820521
```

```
rfecv Adjusted R² on the test set: 0.4157832955818095
```

#### 4) Lasso and Ridge regression

Ridge

Ridge(alpha=0.005)

```
print(reg.coef_)
```

```
[[ 0.24368559  0.00136115  0.80578882 -0.02308458  0.37265904  0.02261838
    0.00612593 -0.0580463  -0.02850832]]
```

```
adjusted_r2_train = adjusted_r2_scorer(reg, X_train, y_train)
adjusted_r2_test = adjusted_r2_scorer(reg, X_test, y_test)

print(f"Ridge Adjusted R2 on the training set: {adjusted_r2_train}")
print(f"Ridge Adjusted R2 on the test set: {adjusted_r2_test}")
```

Ridge Adjusted R<sup>2</sup> on the training set: 0.851752973933268

Ridge Adjusted R<sup>2</sup> on the test set: 0.3980786478053212

Lasso

Lasso(alpha=0.01)

```
print(np.round(reg.coef_, 4))
```

```
[ 0.2435  0.          0.7864 -0.0028  0.3531  0.0019  0.         -0.038  -0.        ]
```

```
adjusted_r2_train = adjusted_r2_scorer(reg, X_train, y_train)
adjusted_r2_test = adjusted_r2_scorer(reg, X_test, y_test)

print(f"sfs Adjusted R2 on the training set: {adjusted_r2_train}")
print(f"sfs Adjusted R2 on the test set: {adjusted_r2_test}")
```

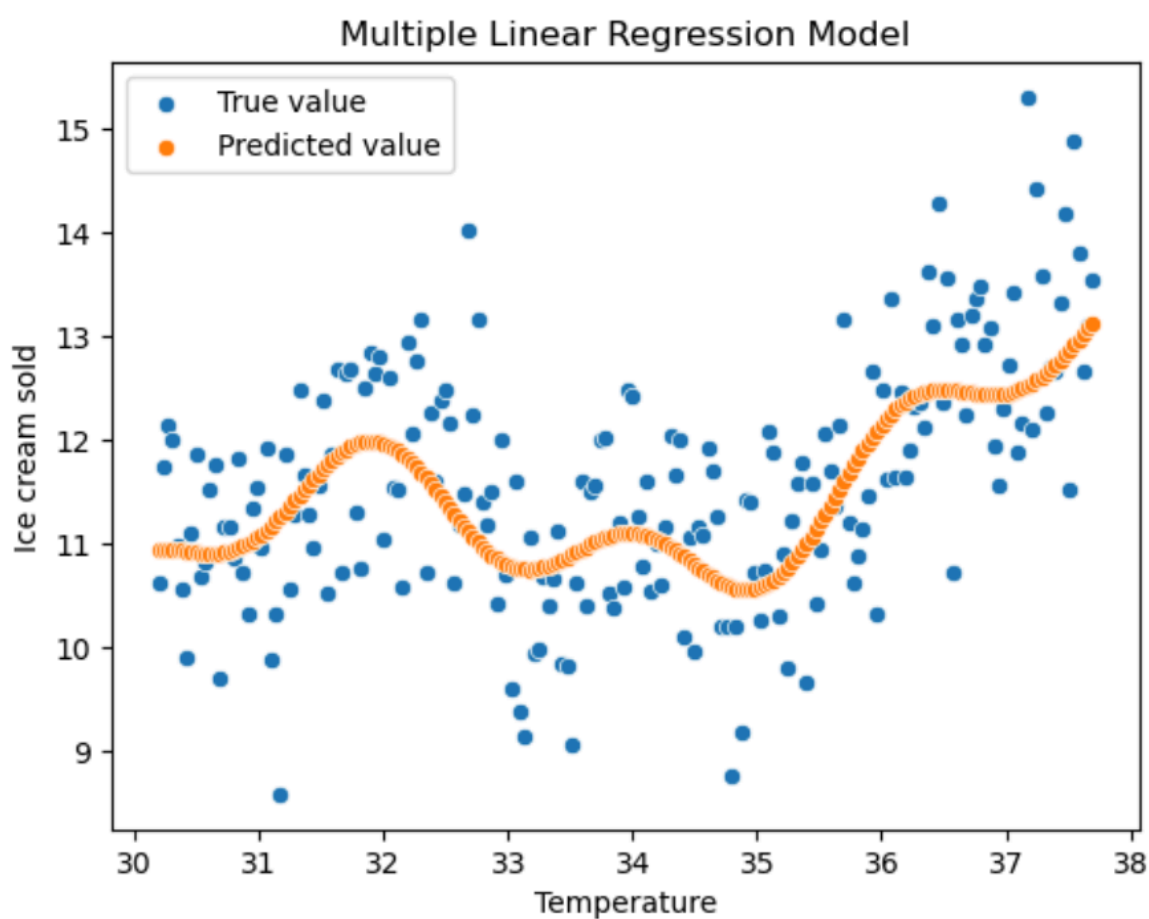
sfs Adjusted R<sup>2</sup> on the training set: 0.851544576269941

sfs Adjusted R<sup>2</sup> on the test set: 0.3938389985858648

The adjusted R<sup>2</sup> scores for Ridge and Lasso regression models on both the training and test sets appear to be lower than the scores achieved previously using SFS and RFECV.

## 5) Model prediction vs real value

	X	y	y_pred	residuals	...	...	...	...	...
0	30.189480	10.632211	10.951414	0.319203	195	37.548165	14.873723	12.918955	-1.954768
1	30.227217	11.743522	10.951832	-0.791690	196	37.585902	13.807789	12.970946	-0.836843
2	30.264954	12.148739	10.949863	-1.198876	197	37.623639	12.669681	13.023379	0.353698
3	30.302690	11.999377	10.945968	-1.053409	198	37.661376	13.106141	13.075733	-0.030408
4	30.340427	10.991670	10.940629	-0.051041	199	37.699113	13.544400	13.127476	-0.416924



The model predictions, represented by the orange points, follow the general trend of the data but do not fully capture its variability. Adjusted  $R^2$  can be used to quantify the extent to which the model explains the variability in ice cream sales. Previously, the RFECV model demonstrated an Adjusted  $R^2$  of approximately 42%, indicating that it explains about 42% of the variability in the data.