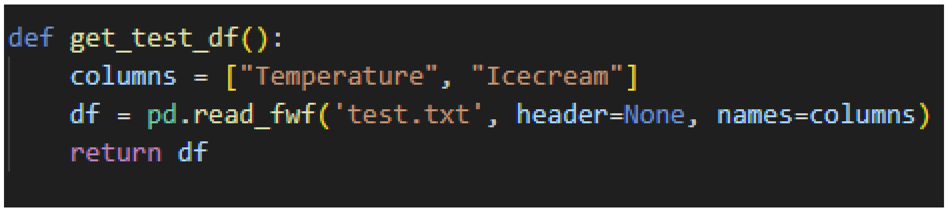
Dereck Taverene
MTH 4330

Statistical Analysis Report

This report explores an ice cream sales dataset with the purpose of creating a predictive model that can effectively forecast future sales using machine learning techniques. This paper seeks to determine the best predictive model for this dataset by comparing linear regression models, feature engineering approaches, and Lasso and Ridge regression methods.

(1) The study begins by looking at the dataset, which consists of two files: *train.txt* and *test.txt*. Each file has two columns: the first column gives quantitative values for *Temperature*, the independent variable, and the second column records the *Number of Ice cream sold* at each temperature. The visual representation of the training dataset is from the output of the Python function *get_DataFrame()*:



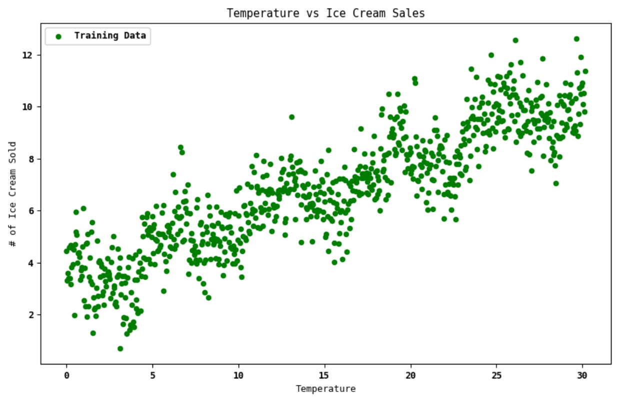
```
def get_test_df():
    columns = ["Temperature", "Icecream"]
    df = pd.read_fwf('test.txt', header=None, names=columns)
    return df
```

Total rows: 800

Index	**Temperature**	**Ice Cream Sold**
0	0.000000	4.456329
1	0.037737	3.301815
2	0.075474	3.595955
3	0.113211	3.401495
4	0.150947	4.571147
...	...	...
795	30.000795	10.928634
796	30.038532	10.106108
797	30.076268	10.521322
798	30.114005	9.814676
799	30.151742	11.364662

Dereck Taverene
MTH 4330

To get a better idea of the relationship within the data, the following figure is provided:

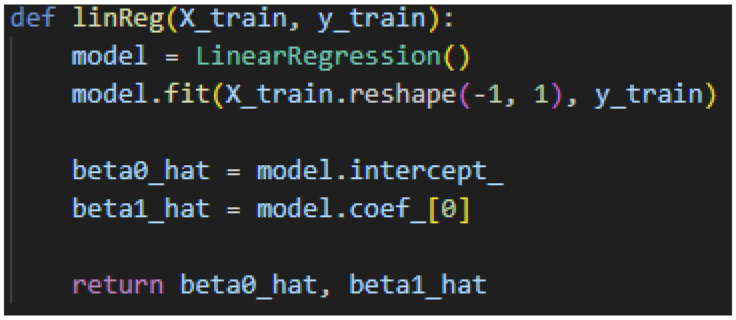


The scatter plot depicts the association between *Temperature (x-axis)* and the *Number of Ice Creams Sold (y-axis)* in the training dataset. The plot indicates a positive association, indicating that as temperatures rise, so does the quantity of ice cream sold. This trend suggests that temperature is a good predictor of Ice Cream Sales. However, the spread of the data points around the overall trend line shows some variance, meaning a linear model might not be the best predictive model.

(2) The association may be examined in more detail by fitting a linear regression model to the data. It is possible to determine whether a linear trend accurately represents the data by determining the model coefficients (betas) and performing hypothesis testing.

Procedure:

1. Fit the training data to the Linear Regression model.
2. Determine the estimated coefficients, β0 and β1, by utilizing model.intercept_ and model.coef_[0] functions. This is done through the Python function *linReg(X_train, y_train)*.



```
def linReg(X_train, y_train):
    model = LinearRegression()
    model.fit(X_train.reshape(-1, 1), y_train)

    beta0_hat = model.intercept_
    beta1_hat = model.coef_[0]

    return beta0_hat, beta1_hat
```

   a. These coefficients represent the optimal values that minimize the Ordinary Least Squares (OLS) error.

Dereck Taverene
MTH 4330

The resulting beta values are:

- β0 (intercept): 3.191037800253056
- β1 (slope): 0.238397630459365

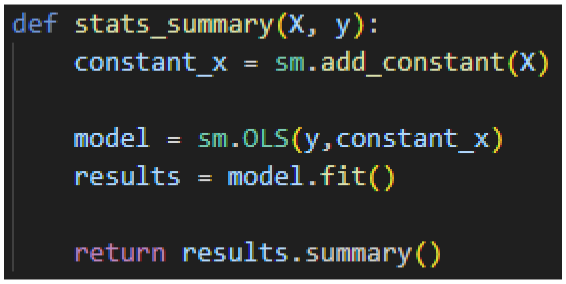
Since β1 ≈ 0 (when rounded to the nearest whole number), indicates no significant linear association between *Temperature* and *Ice Cream Sold*, suggesting that only focusing on *Temperature* as our predictor may not be the most effective way of predicting Ice Cream Sales.

To confirm whether or not there is a linear trend in the data, the following hypothesis is tested:

Confidence Interval: 95%

- H_0: β1 = 0 (no linear trend)
- HA: β1 != 0 (linear trend exists)

To test this hypothesis, the Ordinary Least Squares (OLS) regression results are examined. From the function *stats_summary(X, y)*.

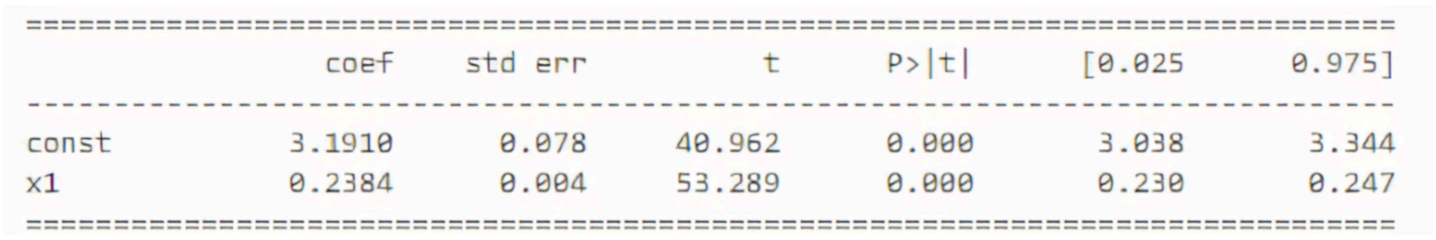


```python
def stats_summary(X, y):
    constant_x = sm.add_constant(X)

    model = sm.OLS(y,constant_x)
    results = model.fit()

    return results.summary()
```

Output:

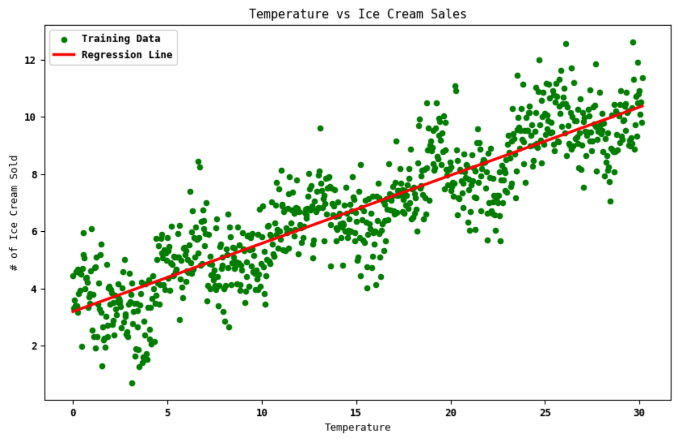


```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.1910      0.078     40.962      0.000       3.038       3.344
x1             0.2384      0.004     53.289      0.000       0.230       0.247
==============================================================================
```

The p-value needed is the one corresponding to *X1* because *X1* corresponds with β1. The p-value is 0.000, which is less than the significance level of 0.05, the null hypothesis can be rejected. Therefore, it is concluded that β1 != 0, indicating a linear trend in the data. It can also be concluded that β0 != 0 since the p-value for *const* is 0.000 too.

Dereck Taverene
MTH 4330

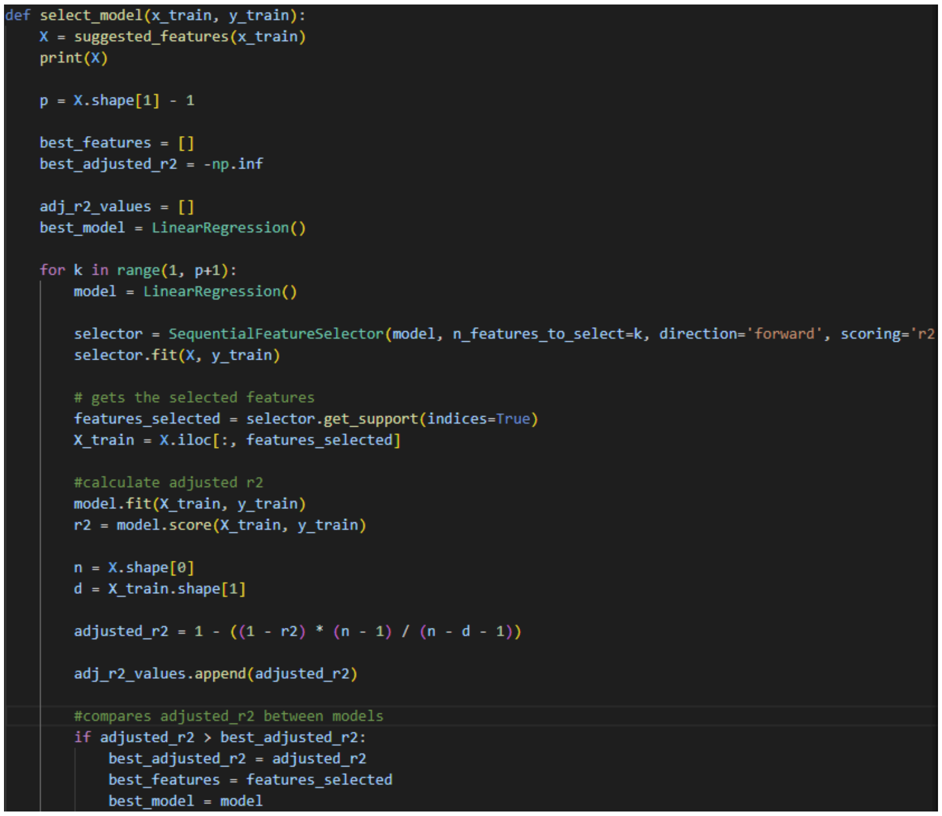
This reveals a linear trend in the data:



but, as previously stated, the plot *Temperature* vs *Ice Cream Sales* shows variability around the trend line, implying that temperature may only have a minor impact on sales.

(3) Due to this fact, *Temperature* shouldn't be the only predictor. In this way, the model can be improved by adding non-linear features. *Cos(x), log(x), cos(4x), sin(3x), sin(5x), and sin(2x)×cos(2x)* are non-linear characteristics that, according to an economist colleague, might enhance the performance of the current linear model.

To select the best features among the suggested features forward selection will be used. forward selection reduces overfitting by using just the most significant predictors and is computationally faster on big datasets than backward selection, it will be used to select the best feature among the features suggested. Because backward selection can be ineffective if many features are irrelevant, backward selection will not be used.



```python
def select_model(x_train, y_train):
    X = suggested_features(x_train)
    print(X)

    p = X.shape[1] - 1

    best_features = []
    best_adjusted_r2 = -np.inf

    adj_r2_values = []
    best_model = LinearRegression()

    for k in range(1, p+1):
        model = LinearRegression()

        selector = SequentialFeatureSelector(model, n_features_to_select=k, direction='forward', scoring='r2
        selector.fit(X, y_train)

        # gets the selected features
        features_selected = selector.get_support(indices=True)
        X_train = X.iloc[:, features_selected]

        #calculate adjusted r2
        model.fit(X_train, y_train)
        r2 = model.score(X_train, y_train)

        n = X.shape[0]
        d = X_train.shape[1]

        adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - d - 1))

        adj_r2_values.append(adjusted_r2)

        #compares adjusted_r2 between models
        if adjusted_r2 > best_adjusted_r2:
            best_adjusted_r2 = adjusted_r2
            best_features = features_selected
            best_model = model
```

Dereck Taverene
MTH 4330

The function *select_model(X_train, y_train)* selects and engineers features according to the procedure below.

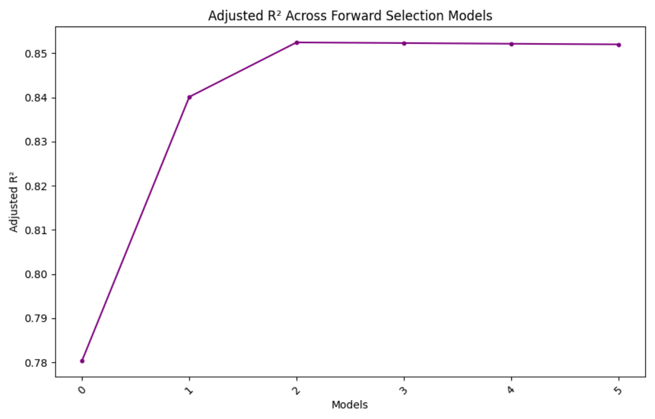
Procedure:

1. Take the values from *Temperature* and plug them into each feature to get the values for those features

	x	cos(x)	log(x)	cos(4x)	sin(3x)	sin(5x)	sin(2x)cos(2x)
0	0.000000	1.000000	-18.420681	1.000000	0.000000	0.000000	0.000000
1	0.037737	0.999288	-3.277118	0.988629	0.112969	0.187567	0.075187
2	0.075474	0.997153	-2.583971	0.954775	0.224491	0.368475	0.148665
3	0.113211	0.993599	-2.178506	0.899207	0.333140	0.536305	0.218762
4	0.150947	0.988629	-1.890824	0.823190	0.437523	0.685097	0.283883
..	...	...	...	...	...	...	...
795	30.000795	0.155037	3.401224	0.812331	0.892925	-0.712091	0.291599
796	30.038532	0.192198	3.402481	0.715395	0.836350	-0.567764	0.349360
797	30.076268	0.229085	3.403736	0.602193	0.769069	-0.403288	0.399175
798	30.114005	0.265647	3.404990	0.475293	0.691939	-0.224493	0.439914
799	30.151742	0.301830	3.406243	0.337584	0.605951	-0.037728	0.470648

2. Loop from 0 to p−1, where p is the number of features.
3. Use the *SequentialFeatureSelector()* function with forward selection specified to compare all possible models with subsets of 3 out of 7 features, based on R-squared. This function is an efficient approach for forward selection, helping identify the best model within that subset.
4. Fit a linear regression model using the selected subset of features, then calculate the adjusted R-squared for that model. Adjusted R-squared is used instead of R-squared because it accounts for the number of predictors, providing a more accurate comparison across models of different dimensions (since additional predictors can artificially inflate R-squared (overfitting)).
5. Return the model with the highest adjusted R-squared and its selected features.

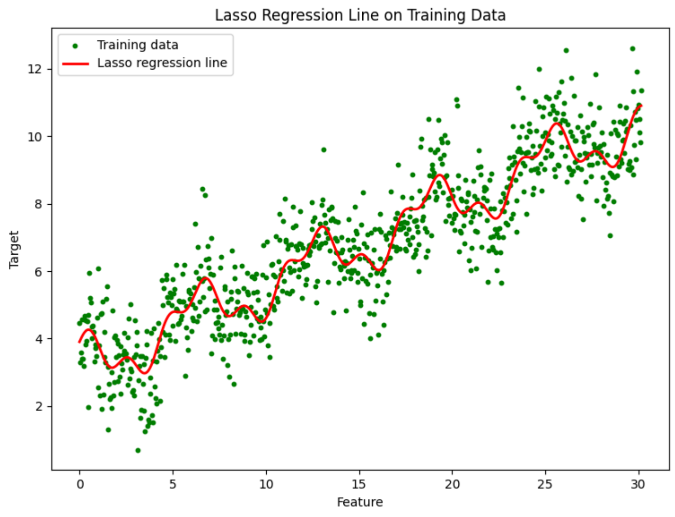
Here is a graph of the what happened in the *select_features(X_train, y_train)*



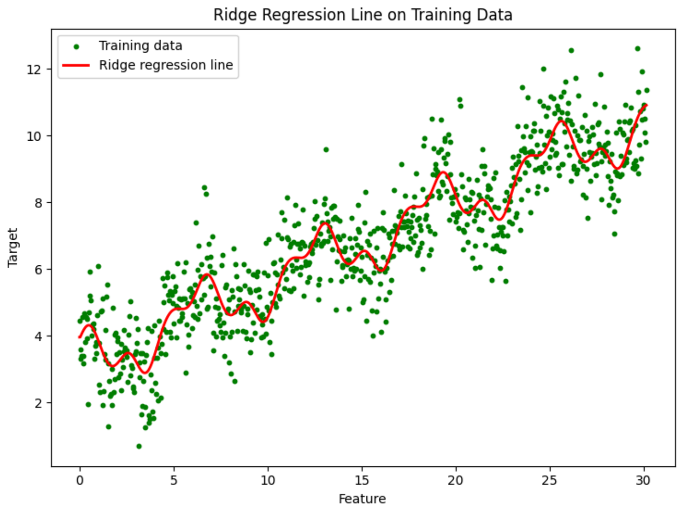
The plot indicates that adding 2 features yields the greatest adjusted R-squared. The selected features *x*, *cos (x)*, and *sin(3x)* were picked based on their performance in the *select_features(X_train, y_train)* function, which identified them as the most effective subset for optimizing model performance.

(4) The next step is to analyze the data using Lasso and Ridge regression models. Each model's performance will be evaluated using R-squared, which takes into consideration the number of predictors and provides a more trustworthy measure when comparing models of varying complexity. Adjusted R-squared is especially relevant in this case because Lasso and Ridge use different regularization strategies that affect the predictors differently. Lasso regression selects features by reducing some coefficients to zero, thereby eliminating less significant predictors, whereas Ridge regression reduces all coefficients proportionally but does not delete any predictors. By comparing modified R-squared values, we may conclude which model provides the best balance of fit and complexity for this dataset. Below are the following two Regressions with their associated adjusted R-squared

Adjusted R-squared: 0.8514336408596881



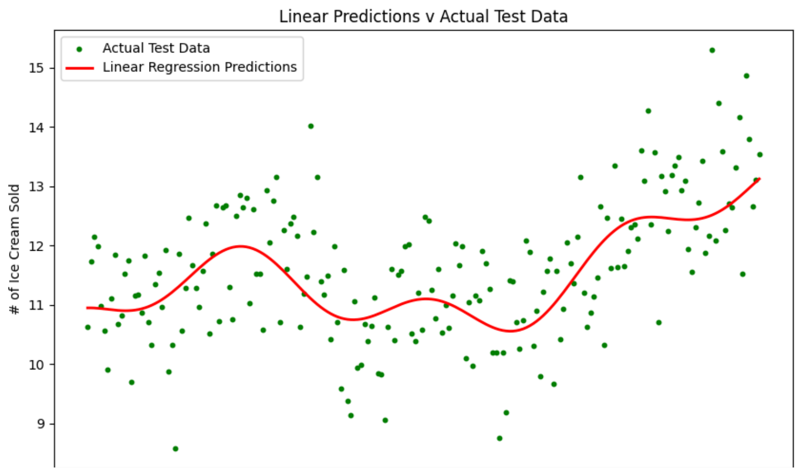
Adjusted R-squared: 8517723399317891



Dereck Taverene
MTH 4330

The corrected R-squared values for the Lasso and Ridge regression models are comparable, demonstrating that both models are efficient in terms of balancing fit and complexity. However, the feature selection model had a significantly better-adjusted R-squared of 0.8524, beating the Lasso and Ridge models by around 0.01. This shows that the feature selection approach may yield a marginally better model performance than regularized regression methods in this scenario.

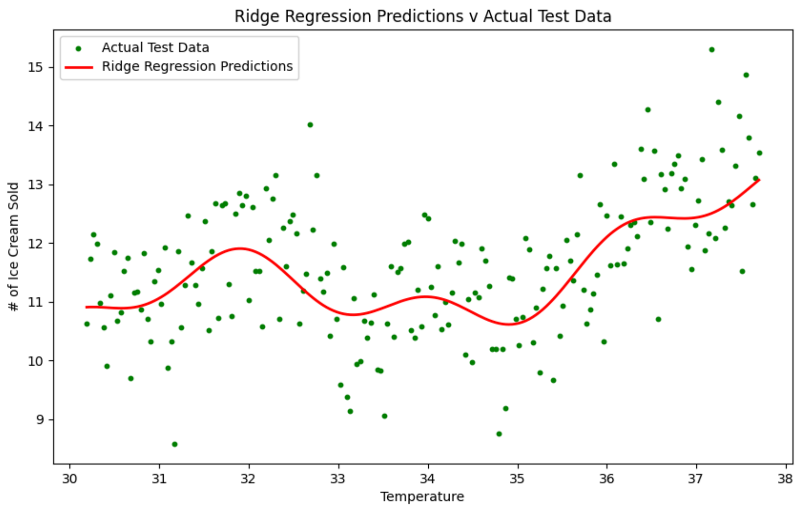
(5) As seen, these three models perform well. The next step is to compare them and assess their practical effectiveness based on their performance on the test data.



Linear Regression Performance on the Test Data:

Mean Squared Error (MSE): 0.7831

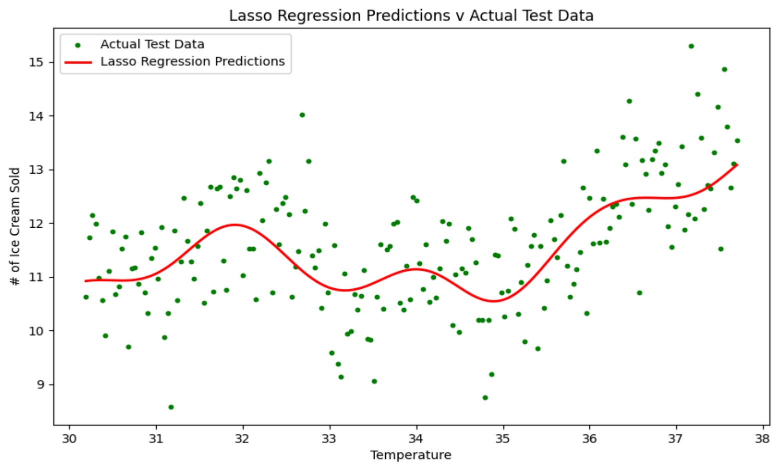
Adjusted R-squared: 0.4158



Ridge Regression Performance on the Test Data:

Mean Squared Error (MSE): 0.7890

Adjusted R-squared: 0.3991



Lasso Regression Performance on the Test Data:

Mean Squared Error (MSE): 0.7797

Adjusted R-squared: 0.4062

Dereck Taverene
MTH 4330

The above figures show how the Lasso, Ridge, and linear regression models' predictions and actual ice cream sales compare. Since Lasso and Ridge reduce some data to zero, and linear regression incorporates all three features, adjusted R-squared is utilized for model comparison. R-squared by itself wouldn't be viable. MSE shows how well each model performed on the test data, the lower the value the higher the accuracy.

Based on the adjusted R-squared and MSE, the Lasso regression model has the highest accuracy on test data, but the linear regression model fits the data the best overall. However, as shown by an MSE of roughly 0.7 for all models, none of them fit the test data very well. This MSE shows that there is high error between the predictors and true values. In addition, an adjusted R-squared of roughly 0.4 across models means that roughly 40% of the variance in the test data is explained. These metrics suggest that while the models provide some predictive ability, further optimization such as feature reduction, could potentially improve model performance.