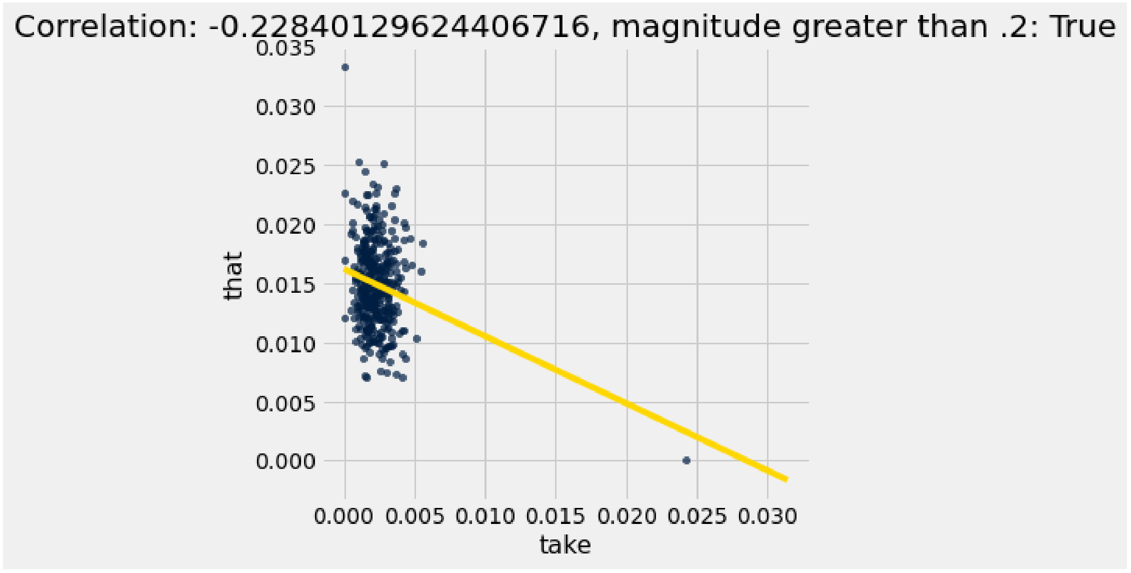
Question 1.2.2. Choose two *different* words in the dataset with a correlation higher than 0.2 or smaller than -0.2 that are not *outer* and *space* and plot a scatter plot with a line of best fit for them. The code to plot the scatter plot and line of best fit is given for you, you just need to calculate the correct values to `r`, `slope` and `intercept`.

Hint 1: It's easier to think of words with a positive correlation, (i.e. words that are often mentioned together).

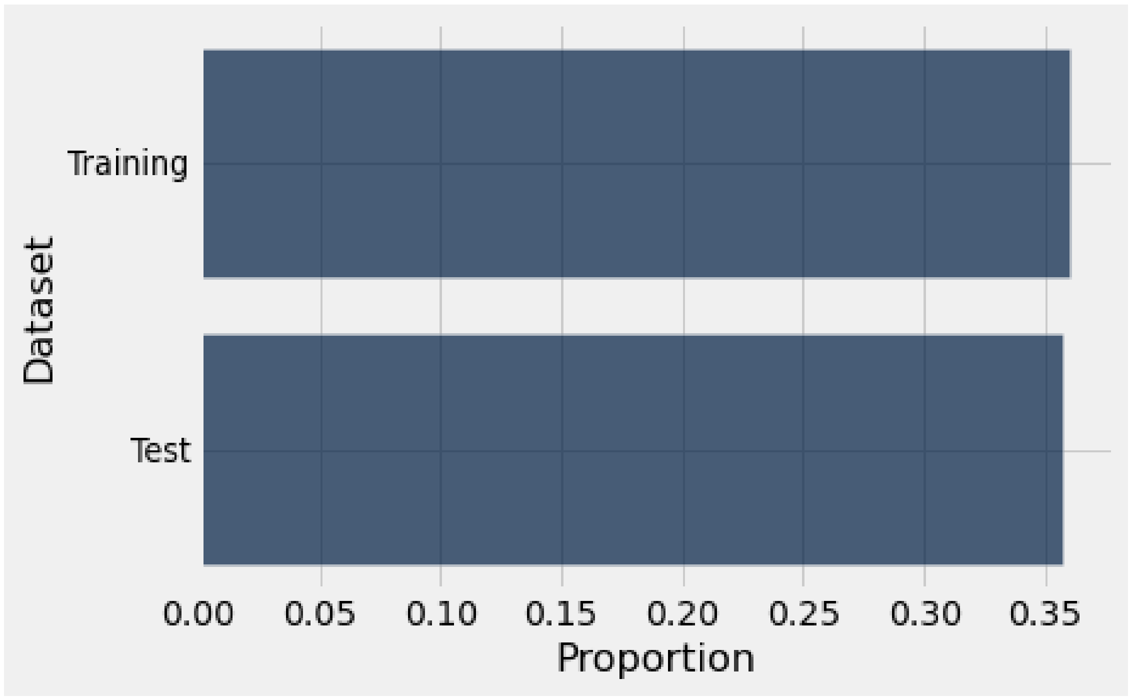
Hint 2: Try to think of common phrases or idioms.

```
In [19]: word_x = "take"
         word_y = "that"

         # These arrays should make your code cleaner
         arr_x = movies.column(word_x)
         arr_y = movies.column(word_y)

         x_su = standard_units(arr_x)
         y_su = standard_units(arr_y)

         r = np.mean(x_su * y_su)

         slope = r * np.std(arr_y) / np.std(arr_x)
         intercept = np.mean(arr_y) - slope * np.mean(arr_x)

         # Don't change these lines of code
         movies.scatter(word_x, word_y)
         max_x = max(movies.column(word_x))
         plots.title(f"Correlation: {r}, magnitude greater than .2: {abs(r) >= 0.2}")
         plots.plot([0, max_x * 1.3], [intercept, intercept + slope * (max_x*1.3)], color = 'gold');
```



Question 1.3.1. Draw a horizontal bar chart with two bars that show the proportion of Comedy movies in each dataset. Complete the function `comedy_proportion` first; it should help you create the bar chart.

```
In [21]: def comedy_proportion(table):
         # Return the proportion of movies in a table that have the Comedy genre.
             return sum(table.column("Genre") == "comedy") / table.num_rows
         # The staff solution took multiple lines. Start by creating a table.
         # If you get stuck, think about what sort of table you need for barh to work ..."""";

         bar_table = Table().with_columns(
             "Dataset", make_array("Training", "Test"),
             "Proportion", make_array(comedy_proportion(train_movies), comedy_proportion(test_movies))
         )
         bar_table.barh("Dataset")
```



0.0.1 Question 3.1.7.

In two sentences or less, describe how you selected your features.

I selected my features by picking words that were common but had strong connotations to them. For example, love has strong positive connotations, making it seem like it would come from a comedy, while dead has strong negative connotations more closely associated with thrillers.

0.0.2 Question 3.3.3.

Do you see a pattern in the types of movies your classifier misclassifies? In two sentences or less, describe any patterns you see in the results or any other interesting findings from the table above. If you need some help, try looking up the movies that your classifier got wrong on Wikipedia.

The classifier misclassifies thrillers and comedies equally, but it seems that a lot of the movies it gets wrong are older movies in the 1930s-1950s. Most of the movies misclassified were released before 2000.

0.0.3 Question 4.1.

Develop a classifier with better test-set accuracy than `classify_feature_row`. Your new function should have the same arguments as `classify_feature_row` and return a classification. Name it `another_classifier`. Then, check your accuracy using code from earlier.

You can use more or different features, or you can try different values of `k`. (Of course, you still have to use `train_movies` as your training set!)

Make sure you don't reassign any previously used variables here, such as `proportion_correct` from the previous question.

```
In [64]: # To start you off, here's a list of possibly-useful features
         # Feel free to add or change this array to improve your classifier
         new_features = make_array("laugh", "marri", "dead", "heart", "cop",
                                   'love', 'fun', 'scream', 'joke', 'fight',
                                   'cry', 'terrifi', 'break', 'kill', 'monster')
         train_new = train_movies.select(new_features)
         test_new = test_movies.select(new_features)

         def another_classifier(row):
             return classify(row, train_new, train_movies.column(3), 13)

         test_guesses = test_new.apply(another_classifier)
         proportion_correct = sum(test_movies.column(3) == test_guesses) / test_movies.num_rows
         print(proportion_correct)

         correct = test_movies.column(3) == test_guesses
         test_movie_correctness = Table().with_columns("Title", test_movies.column(0), "Genre", test_mo
         test_movie_correctness.sort('Was correct', descending = False).show()
```

```
0.7321428571428571
```

```
<IPython.core.display.HTML object>
```

0.0.4 Question 4.2.

Do you see a pattern in the mistakes your new classifier makes? What about in the improvement from your first classifier to the second one? Describe in two sentences or less.

Hint: You may not be able to see a pattern.

There wasn't really much of a pattern. The movies are all from different time periods and the classifier didn't misclassify a genre more than another one.

0.0.5 Question 4.3.

Briefly describe what you tried to improve your classifier.

I tried to use more specific words that were more likely to strongly appear in one genre than another one.