4. **Interpreting the difference in performance between the models:** To understand why the model with all word counts performs better than the one with only the *selected_words*, we will now examine the reviews for a particular product.

- We will investigate a product named *'Baby Trend Diaper Champ'*. (This is a trash can for soiled baby diapers, which keeps the smell contained.)
- Just like we did for the reviews for the giraffe toy in the Jupyter Notebook in the lecture video, before we start our analysis you should select all reviews where the product name is *'Baby Trend Diaper Champ'*. Let's call this table *diaper_champ_reviews*.
- Again, just as in the video, use the *sentiment_model* to predict the sentiment of each review in *diaper_champ_reviews* and sort the results according to their *'predicted_sentiment'*.
- What is the *'predicted_sentiment'* for the most positive review for *'Baby Trend Diaper Champ'* according to the sentiment_model from the Jupyter Notebook from lecture? ***Save this result to answer the quiz at the end.***
- Now use the *selected_words_model* you learned using just the *selected_words* to predict the sentiment most positive review you found above. *Hint: if you sorted the diaper_champ_reviews in descending order (from most positive to most negative), this command will be helpful to make the prediction you need:*

```
selected_words_model.predict(diaper_champ_reviews[0:1], output_type='probability')
```

Save this result to answer the quiz at the end.

- Why is the *predicted_sentiment* for the most positive review found using the model with all word counts (*sentiment_model*) much more positive than the one using only the *selected_words (selected_words_model)*? *Hint: examine the text of this review, the extracted word counts for all words, and the word counts for each of the selected_words, and you will see what each model used to make its prediction.* ***Save this result to answer the quiz at the end.***

What is the accuracy of the *selected_words_model* on the *test_data*? What was the accuracy of the *sentiment_model* that we learned using all the word counts in the Jupyter Notebook above from the lectures? What is the accuracy *majority* class classifier on this task? How do you compare the different learned models with the baseline approach where we are just predicting the majority class? ***Save these results to answer the quiz at the end.***

Hint: we discussed the majority class classifier in lecture, which simply predicts that every data point is from the most common class. This is baseline is something we definitely want to beat with models we learn from data.

4. **Interpreting the difference in performance between the models:** To understand why the model with all word counts performs better than the one with only the *selected_words*, we will now examine the reviews for a particular product.

- We will investigate a product named *'Baby Trend Diaper Champ'*. (This is a trash can for soiled baby diapers, which keeps the smell contained.)
- Just like we did for the reviews for the giraffe toy in the Jupyter Notebook in the lecture video, before we start our analysis you should select all reviews where the product name is *'Baby Trend Diaper Champ'*. Let's call this table *diaper_champ_reviews*.
- Again, just as in the video, use the *sentiment_model* to predict the sentiment of each review in *diaper_champ_reviews* and sort the results according to their *'predicted_sentiment'*.
- What is the *'predicted_sentiment'* for the most positive review for *'Baby Trend Diaper Champ'* according to the sentiment_model from the Jupyter Notebook from lecture? ***Save this result to answer the quiz at the end.***
- Now use the *selected_words_model* you learned using just the *selected_words* to predict the sentiment most positive review you found above. *Hint: if you sorted the diaper_champ_reviews in descending order (from most positive to most negative), this command will be helpful to make the prediction you need:*

```
selected_words_model.predict(diaper_champ_reviews[0:1], output_type='probability')
```

Save this result to answer the quiz at the end.

- Train a logistic regression classifier (use *turicreate.logistic_classifier.create*) using just the *selected_words*. Hint: you can use this parameter in the *.create()* call to specify the features used to be exactly the new columns you just created:

```
features=selected_words
```

Call your new model: *selected_words_model*.

- You will now examine the weights the learned classifier assigned to each of the 11 words in *selected_words* and gain intuition as to what the ML algorithm did for your data using these features. In Turi Create, a learned model, such as the selected_words_model, has a field 'coefficients', which lets you look at the learned coefficients. You can access it by using:

```
selected_words_model['coefficients']
```

The result has a column called 'value', which contains the weight learned for each feature.

Using this approach, sort the learned coefficients according to the 'value' column using *.sort()*. Out of the 11 words in *selected_words*, which one got the most positive weight? Which one got the most negative weight? Do these values make sense for you? ***Save these results to answer the quiz at the end.***

3. **Comparing the accuracy of different sentiment analysis model:** Using the method

```
.evaluate(test_data)
```

In our case, if this condition doesn't hold, the count of *'awesome'* should be 0.

Using these tips, you can now write the awesome_count function.

- Next, you will use *.apply()* to iterate *awesome_count* for each row of *products['word_count']* and create a new column called *'awesome'* with the resulting counts. Here is what that looks like:

```
products['awesome'] = products['word_count'].apply(awesome_count)
```

And you are done! Check the *products* SFrame and you should see the new column you just create.

- Repeat this process for the other 11 words in *selected_words*. (Here, we described a simple procedure to obtain the counts for each *selected_word*. There are other more efficient ways of doing this, and we encourage you to explore this further.)
- Using the *.sum()* method on each of the new columns you created, answer the following questions: Out of the *selected_words*, which one is most used in the dataset? Which one is least used? ***Save these results to answer the quiz at the end.***

2. **Create a new sentiment analysis model using only the selected_words as features:** In the Jupyter Notebook above, we used word counts for all words as features for our sentiment classifier. Now, you are just going to use the *selected_words*:

- Use the same train/test split as in the Jupyter Notebook from lecture:

```
train_data,test_data = products.random_split(.8, seed=0)
```

- Train a logistic regression classifier (use *turicreate.logistic_classifier.create*) using just the *selected_words*. Hint: you can use this parameter in the *.create()* call to specify the features used to be exactly the new columns you

We could use a for loop to iterate this logic for each row of the products SFrame, but this approach would be really slow, because the SFrame is not optimized for this being accessed with a for loop. Instead, we will use the *.apply()* method to iterate the the logic above for each row of the *products['word_count']* column (which, since it's a single column, has type SArray). Read about using the .apply() method on an SArray here.

We are now ready to create our new columns:

- First, you will use a Python function to define the logic above. You will write a function called *awesome_count* which takes in the word counts and returns the number of times *'awesome'* appears in the reviews.

A few tips:

i. Each entry of the 'word_count' column is of Python type dictionary.

ii. If you have a dictionary called *dict*, you can access a field in the dictionary using:

```
1  dict['awesome']
```

but only if *'awesome'* is one of the fields in the dictionary, otherwise you will get a nasty error.

iii. In Python, to test if a dictionary has a particular field, you can simply write:

```
1  if 'awesome' in dict
```

In our case, if this condition doesn't hold, the count of *'awesome'* should be 0.

Using these tips, you can now write the awesome_count function.

- Next, you will use *.apply()* to iterate *awesome_count* for each row of *products['word_count']* and create a new

What you will do

Now you are ready! We are going do four tasks in this assignment. There are several results you need to gather along the way to enter into the quiz after this reading.

In the Jupyter notebook above, we used the word counts for all words in the reviews to train the sentiment classifier model. Now, we are going to follow a similar path, but only use this subset of the words:

```
selected_words = ['awesome', 'great', 'fantastic', 'amazing', 'love', 'horrible', 'bad
```

Often, ML practitioners will throw out words they consider "unimportant" before training their model. This procedure can often be helpful in terms of accuracy. Here, we are going to throw out all words except for the very few above. Using so few words in our model will hurt our accuracy, but help us interpret what our classifier is doing.

1. **Use .apply() to build a new feature with the counts for each of the selected_words:** In the notebook above, we created a column 'word_count' with the word counts for each review. Our first task is to create a new column in the products SFrame with the counts for each selected_word above, and, in the process, we will see how the method .apply() can be used to create new columns in our data (our features) and how to use a Python function, which is an extremely useful concept to grasp!

Our first goal is to create a column *products['awesome']* where each row contains the number of times the word *'awesome'* showed up in the review for the corresponding product, and 0 if the review didn't show up. One way to do this is to look at the each row *'word_count'* column and follow this logic:

- If *'awesome'* shows up in the word counts for a particular product (row of the products SFrame), then we know how often *'awesome'* appeared in the review,
- if *'awesome'* doesn't appear in the word counts, then it didn't appear in the review, and we should set the count for *'awesome'* to 0 in this review.

We could use a for loop to iterate this logic for each row of the products SFrame, but this approach would be really slow, because the SFrame is not optimized for this being accessed with a for loop. Instead, we will use the *.apply()* method to iterate the the logic above for each row of the *products['word_count']* column (which, since it's a single

8. **How do you compare the different learned models with the baseline approach where we are just predicting the majority class?** — 1 / 1 point

Correct

9. **Which of the following ranges contains the *'predicted_sentiment'* for the most positive review for *'Baby Trend Diaper Champ'*, according to the *sentiment_model* from the IPython Notebook from lecture?** — 1 / 1 point

Correct

10. **Consider the most positive review for *'Baby Trend Diaper Champ'* according to the *sentiment_model* from the IPython Notebook from lecture. Which of the following ranges contains the predicted_sentiment for this review, if we use the *selected_words_model* to analyze it?** — 1 / 1 point

Correct

11. **Why is the value of the *predicted_sentiment* for the most positive review found using the *sentiment_model* much more positive than the value predicted using the *selected_words_model*?** — 1 / 1 point

Correct

4. **Out of the 11 words in *selected_words*, which one got the most negative weight in the *selected_words_model*?** **1 / 1 point**

   (Tip: when printing the list of coefficients, make sure to use print_rows(rows=12) to print ALL coefficients.)

   Correct

5. **Which of the following ranges contains the accuracy of the *selected_words_model* on the *test_data*?** **1 / 1 point**

   Correct

6. **Which of the following ranges contains the accuracy of the *sentiment_model* in the IPython Notebook from lecture on the *test_data*?** **1 / 1 point**

   Correct

7. **Which of the following ranges contains the accuracy of the majority class classifier, which simply predicts the majority class on the *test_data?*** **1 / 1 point**

   Correct

8. **How do you compare the different learned models with the baseline approach where we are just predicting** **1 / 1 point**

Congratulations! You passed!

Grade received 100%	Latest Submission Grade 100%	To pass 80% or higher

Go to next item

1. **Out of the 11 words in *selected_words*, which one is most used in the reviews in the dataset?** — 1 / 1 point

   Correct

2. **Out of the 11 words in *selected_words*, which one is least used in the reviews in the dataset?** — 1 / 1 point

   Correct

3. **Out of the 11 words in *selected_words*, which one got the most positive weight in the *selected_words_model*?** — 1 / 1 point

   (Tip: when printing the list of coefficients, make sure to use print_rows(rows=12) to print ALL coefficients.)

   Correct

4. **Out of the 11 words in *selected_words*, which one got the most negative weight in the *selected_words_model*?** — 1 / 1 point