Like this course? Become an expert by joining the Machine Learning Specialization (/specializations/machine-learning).

Upgrade

Course Home (/learn/ml-foundations/home/welcome) ❯ Week 3 (/learn/ml-foundations/home/week/3) ❯ F

# Analyzing product sentiment

In this module, we focused on classifiers, applying them to analyzing product sentiment, and understanding the types of errors a classifier makes. We also built an exciting IPython notebook for analyzing the sentiment of real product reviews.

In this assignment, we are going to explore this application further, training a sentiment analysis model using a set of key polarizing words, verify the weights learned to each of these words, and compare the results of this simpler classifier with those of the one using all of the words. These techniques will be a core component in your capstone project.

Follow the rest of the instructions on this page to complete your program. When you are done, *instead of uploading your code, you will answer a series of quiz questions* (see the quiz after this reading) to document your completion of this assignment. The instructions will indicate what data to collect for answering the quiz.

## Learning outcomes

- Execute sentiment analysis code with the IPython notebook

- Load and transform real, text data

- Using the *.apply()* function to create new columns (features) for our model

- Compare results of two models, one using all words and the other using a subset of the words

- Compare learned models with majority class prediction

- Examine the predictions of a sentiment model

- Build a sentiment analysis model using a classifier

Help Center

## **Resources you will need**

You will need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1.

## Download the data and starter code

Before getting started, you will need to download the dataset and the starter IPython notebook that we used in the module.

- Download the product review dataset here in SFrame format: amazon_baby.gl.zip (https://eventing.coursera.org/api/redirectStrict/5De7n3_qDNl0xwT98_P2YFxT6Ho4OwhReK2aQ GXw219dJD9VUg6ODQHfipf4D8suH8ozXOeiCw0kmcCDArBE0w.qrmsv8sDY-q6ZSDHCy386w.tQQbdZHlivw24_eW8xxG4uxePDoGKcF020IjSLQwUeRUl2JqwWYa1deBAhh0_r_qAnKYyqhjfGYseQLE_fuVx0EDbQr7SB92WQhoMva5dRbNgjc0S8sfJgCUd_-IrdT03hDtLOniXSqoC0jAj1p3Ij-O2BdknV2iyqESRotmiT2gMWaxWX6Ua7UOiVkX5BiI9eDYPCLB7qfEDVGprzAXKRfmkgWgRqT1 Sp7AfLxD-FLpnxzPmK56MVEWB_XJM85XIMdasTuVsmrf5TyeqgsTtmiG62MISXjZZm0u2zdLI7bG9Slz_0O Z77XMjN26k7lOn4aXSUvFEDlZpSo4uCq3LgtzBopm2Rv7WT-5SmcplpJTJSpB3KL5x-ewYGT1T5naaP3bMd6TgHKL3tMTmv51QwuCaBPxjz6JPozF026aYDtJKLTmgTefrUW9guW_1tT s)

- Download the sentiment analysis notebook from the module here: Analyzing product sentiment.ipynb (https://eventing.coursera.org/api/redirectStrict/UgBsyH3dmE56TTLxti7VoZQbmfb0fR9AWcTTt_5f_9QfbNiRRr_wGY_blGCOM9JkqoHHS-7OmBRhI4OZj2tX5w.gb9AVtOns8ZWF3acai515w.z-6ZW1OyZXa2fNL3fzi0ZkpAmF-dorMklsOcTjPku-wKWgJlX5Yc4-icQmYNB2FLxOU3Qtdng6HyHZxEE42awOO4_mdfUWQYGKpkkz4Uq2wC3FUOPluytw-jsHwGWw3jih6btjw7-VSjKm5GvynfFb9D7T6gvcpfScC1vtS6vy5fQK0Z9M-cr8Sp9KaXmaHi_40x4vMKWSwbqJiZvcKmFbmxn1foRofx8t96Y8HvuWWEEgrvO18yJyJtirhwtqj-26m7y6FB-aNrcwiHErH5jCoDuSmujHmMGmdFBbK-V9TJmQKXW45HOQKwJ_B1kvR4BCkegLOlLiBWmrGGg0WfKLaunv9qL_xgXaVUEJ0W9f_yExp-WalfX513yeexXC-G1s732xxRFSg0-tHkcZgH8KU9vmWK3vhdfXTo9hqo4MzG77sd6D029g8GyLhv5LGw32zWtKpTmRmP4PSfUln9o9 z57NNJ9ro_2r-lpDxoIVS77JPe68bt9f1Fr5GlUgVU)

- Save both of these files in the same directory (where you are calling IPython notebook from) and unzip the data file.

Now you are ready to get started!

## *Note: If you would rather use other ML tools...*

You are welcome to use any ML tool for this course, such as scikit-learn (https://eventing.coursera.org/api/redirectStrict/_7wSdUaC2gycPotnqiQ8AvSutoebw6W0yIpcXJe79x XgzyGtqgYjGDbVLu8hxYT9TlpiFPE31bFSPbVwPyGfkQ.6Z7iepryWC1E8JOFv8LlBw.KnUA2aFlYUjZ MyMAibi4ccOR78LZfloX4mvF4aRIDIqW51VJnshNLPisIyLQE2iScSGh0bK4dAPESQM6kYlJ7BoIccu 1zxUNepdd1qKmaCyluO00cWbMnlBXPKveEVYLNkBSfPC5LIhDjU-ocNeQihBSSwkpXK1T0HtOp2UHU9QWapacjz6nIykzGVzoQWruKZekt32XeHNl4n3qFWijpKPACElhY dxXHUnYwFoOMqH7lzLIQ-sa2ydf4YUMT0Ulw0wsrSjaivLBQ3vn-eUu62ILEfn9ALy4KNGY16az0jM). Though, as discussed in the intro module, *we strongly recommend you use IPython Notebook and GraphLab Create. (GraphLab Create is free for academic purposes.)*

If you are choosing to use other packages, we still recommend you use SFrame, which will allow you to scale to much larger datasets than Pandas. (Though, it's possible to use Pandas in this course, if your machine has sufficient memory.) The SFrame package is available in open-source under a permissive BSD license (https://eventing.coursera.org/api/redirectStrict/6sQyDxhRkx_RLfyOGjClCX475afAnAO0yEOo41iPo CZcjWOIka4Nri5Z-A6x_1HnrbeW9LhzBhLSDardIS9Lag.SGAE4oVvpcTDSKUZr_xB0Q.TMQZ1bmdJzY2N58-gsIvbIcpPkda5co42EVX5letzHbd_0pEw2U2HmJXt30qpcYRehFygLOvWO4ZlNklLW-gtLuWl_rEBAwv9kkmOAUdzpJrO_UH98sctTKA7K1pZt6B4gO7F0T9d5M2oGEmsleuMLXMKOCKCT LhxAZhLslCE6x1ceclsaQj4Cp4hMPyozPXnFMVul0Xc7Z7N1v0D72weyuCSsbDZoFAP8lWYKQkjY00 Fr4rsTnykxWPRIiDT6kPEkpwZDb0hVCaMYsaUcjEre-a4Z-G3po9UmOl8dlIktQ). So, you will always be able to use SFrames for free.

If you are not using SFrame, here is the dataset for this assignment in CSV format, so you can use Pandas (https://eventing.coursera.org/api/redirectStrict/JhsFtJ6-uhMUKhNHYycKVQwzM9U6u0W5JNsoZchyQfScz8VUSFJBfXCvx3yWtgRLSPmRsRUZXDEOtXkJc0 KU_w.hQmYGOYgCWqjh8xGaEYuwQ.BNxBCcQSauq5iBlsMgmuMduobex4sawdzdH4-4w-SFecyBfo3yMDJPRVB9nGWge8B2MdN49iTvh0GiTqcjRvRg4eqVCAkG2hTQx-ZM3MkAvriWsDjDqAi3lBeJMOHKKElW3wMkzVm6gkCryT2apjWlxOVqxqQh8Wb4HV2m4l7vj-71UCKl1ohc1KLzSu3PFWK_qybtRMc4nhtxWQts1IRhq_TGoMIRmVli3Xb7AF7trwm5lnT7puCxuQbA Gq3hdGZ4dU3aimx3TCuO83J963uA) or other options out there: amazon_baby.csv (https://eventing.coursera.org/api/redirectStrict/HBmUzUIxLIauVWwJN0-F7l-svmPt6_MFQFFjDtELqGo0sSvhsNWbnvyIMDQvkWjQ4aISaRL22R6A_lZbhKGZ3Q.xPaPZvjbi4C0Rl 3GokLk0Q.k_8Jdh7l63Eqp2wrRQFEP8-x-trBqFj02qtah0Iqj1JXS1I_7qHdlVdY8yrUNRuJi-_bEGefGnHprTh6iHtG_czmhkCTbfk5WnZsrfg6nc8KjlZcDVXjHBVPcZV28WE8lQFT1XgZci6JLMLjSu IiJRA_inQ-zUDWrg8bIappCDHFMBq-fqCQiFIBd6zkyk_DN-BySY3a1ValseBOckIFgtHBvxjf8eAziDfY01h-Tx5tFu_luElznceGjxVus38FibXVBiixULQ-zW61X0DPBUoFqs2zrqxS3vTuCGklvFkmlVYLg4VSuxlUFNxFrNMgIaDT1w7lp1gQUDAUnG9_VtYM SpA3DN3m-LjH9CNAxHAhenUfHjvrTnd3iBKiRJGu)

# Watch the video and explore the IPython notebook on analyzing sentiment

If you haven't done so yet, before you start, we recommend you watch the video where we go over the IPython notebook on analyzing product sentiment using classifiers from this module. You can then open up the IPython notebook we used and familiarize yourself with the steps we covered in this example.

## 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 IPython 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', 'ba
d', 'terrible', 'awful', 'wow', 'hate']
```

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 column, has type SArray). Read about using the .apply() method on an SArray here. (https://eventing.coursera.org/api/redirectStrict/dZV0tKdMZ8j69EP04jP8cFLz8vICxMQjSQYkab3f53 89aCNgxnQdmadd2LVfYauXQ1XG1v4WFF_cUQutce2SZg.HpPPmB9MoDv7u_ZJOwUjMA.EyKbSZ u39lvnabRCtqdLulndMHIcJZpJg6W4Zjqs4pVy1iZmSxyzjr40iKqWlO35IzDfstTxu6Xy2_jg2vJvnVFAw NmjZMtqmdvLy_LXkT7A9kQ0YN7srECbYj1lAlpuXe_w6JfI3s4HHw2ljzdb2Vbtvn7KCbvCrdnuh0qQO Vy_hV_NSoSRZKi7q1zm62GZ14D57U7Zj1etiDWJ3RQ_lkwPnXm9zbBN7oxISCIEUcjq9c9HwVg0MIx 7x1yhXi-ZcV13vgWlBfu0MS5MZriXKnBB7uKbPqb0zTzwebwpYE3zywGfB2B3q3Nahr- 9XA0ZL88to4NEbJjdzu-3h0yzE5iRYMhpR0KJ5D4MqQMRsZf928SBek4DvF-1DZnG_oo0p7fK- VeTmmp_s2eK2t9-Kw)

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 (https://eventing.coursera.org/api/redirectStrict/WTQXoJgDP7cazLVxEWM- JL0U4EQRFGkefRQqEf25I_GeEyddzpPjfOEtD4cs8Z_Dvxq- AWKfTEpQLCkI8m0WZA.nR2DPIQNOlHZvBfVZgon6g.9H1jhjHiSayb2BYgIOPpDy71cb6PpsI2W-

DHA0I623mP9HEwkFb1ifaroUZQ6xJwu28TsdZHbyvdTzBvUk6vqpX8ydkjGSbdhdZw0WBZ1w8L9Fq
Xo-vRHgkRs-
KcRR5Wvdl0u5kCnX_90aPERoXD2UzaBJrDpn81oAHxHGQlPuBH42SUgp1fuhROn9G5SClg0LkPZ
gDvmP9J2WSCB7kL-
hy0KwpoXw8Oi7X1c9mSbQMU_EJ7OFKb82KTes5vSdL3TuewzJyFUgPAL_yYFSdJNwv847wt-
F4JfX3JwZTuHf7zfwzT9MUdQeGSutMZsc2I5W5GRhCGA04WFTOt3_IS6ztk_mzcpnLBauu8q-
GPG14rHNBigMnpo6nuHi7qUW_y).

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

```
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:

```
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 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 IPython 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 IPython Notebook from lecture:

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

- Train a logistic regression classifier (use *graphlab.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 GraphLab 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)
```

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 IPython 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 IPython 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 IPython 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.**

---

👍  👎  ⚑

❮ (/learn/ml-
foundations/lecture/D8QvW/exploring-
the-most-positive-negative-aspects-of-
a-product)

❯ (/learn/ml-
foundations/exam/Impum/analyzing-
product-sentiment)