Published in Towards Data Science

Edward Ma Follow
Jul 29, 2018 · 9 min read · Listen

3 ways to interpretate your NLP model to management and customer

Photo: https://cdn.pixabay.com/photo/2017/02/22/20/02/landscape-2090495_960_720.jpg

Machine Learning (ML) and Artificial Intelligence (AI) changed the landscape in doing business. Lots of company hire data scientist (I am one of them:)) to deliver a data product.

It is absolute easy to deliver a engine with "high" accuracy (Here is the reason why I double quoted). If somebody ask about reason, we can say that the model is black box or it is a statistical calculation few year ago but we cannot say that anymore nowadays. We also need to convince product team and management to trust our model but we cannot simply say that the engine "learnt" from data or it is a black box which cannot be explained as well.

Of course, we may not need to explain model in some special scenario. For example, you do not want to explain the fraud detection model to public. Otherwise, public can find a way to "hack" the system.

After reading this article, you will understand:

- What is model interpretation?
- Why do we need to interpretate model?
- Interpreting NLP model

What is model interpretation?

Model interpretation means providing reason and the logic behind in order to enable the accountability and transparency on model. There are many types of model interpretations which including:

- During ***exploratory data analysis*** (EDA), we may apply principal component analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) to understand about the feature
- After ***built model***, we will use various metric to measure classification and regression model. Accuracy, precision and recall used in classification while mean squared error (MSE) used in regression. This kind of metric help us to understand the model performance
- Besides ***performance***, static feature summary such as feature importance can be retrieved from model. However, it only exist in those decision tree base algorithm such as Random Forest and XGBoost.
- When ***evaluating model***, we want to know why we predict it wrongly. We can leverage library to explain it so that we can fix it.

Photo: https://pixabay.com/en/whiteboard-man-presentation-write-849812/

Before explaining the model, we can first understand the criteria of model interpretation:

Interpretability

- Intrinsic: We do not need to train another model to explain the target. For example, it is using decision tree or liner model
- Post hoc: The model belongs to black-box model which we need to use another model to interpret it. We will focus on this area in the following part

Approach

- Model-specific: Some tools are limited to specific model such as liner model and neural network model

Level

- Global: Explain the overall model such as feature weight. This one give you a in general model behavior
- Local: Explain the specific prediction result.

Why do we need to explain model?

As mentioned before, we should explain in most of the scenario because:

- ***Accountability***. Model should has a accountability to provide a correct (relatively) answer to consumer. As a model author, we should validate model features to guarantee it help on make a correct decision but not including as much feature as we can.
- ***Transparency***. Machine learning is not a black box and we should provide the model transparency to customer and management. So that they know why you are doing. Just like open source, people are willing to use open source much more as they know what you are doing.
- ***Trustability***. It is the basic requirement if consumer want to use the classification or prediction result. Most of the time, use has some domain knowledge such as trader need to understand why you provide the decision for buy/ sell particular stock.

Interpreting NLP model

As you know, I mainly focus on NLP. Therefore, I only demonstrate the capability of interpreting NLP model although the following library can also explain other problems such as regression and image.

For the following demonstration, I did NOT do any preprocessing for the sake of keeping it easy to understand. In real life business problem, we should do the preprocessing all the time. Otherwise, garbage in garbage out.

Several libraries helps to explain model which including Explain Like I am Five (ELI5), Local Interpretable Model-agnostic Explanations (LIME) [1] and Skater.

- ***Library:*** I will demonstrate how we can use ELI5 and LIME. How about Skater? Stay tuned, will explain reason why I do not use it in NLP.
- ***Algorithm:*** In the following demonstration, it will include linear model (Logistic Regression in scikit-learn), ensemble model (Random Forest in scikit-learn, XGBoost) and deep learning model(self build word embedding in keras). For word embedding, you can check out my previous post for detail.

ELI5

Photo: http://explain-like-i-am-five.tumblr.com/post/158611418472/eli5-the-concept-of-bankruptcy

ELI5 provides both global model interpretation and local model interpretation. You may simply consider the global model interpretation as a feature importance but it not only support decision tree algorithm such as Radom Forest and XGBoost but also all sci-kit learn

a particular word is not provided. For example, “I like apple”. It will may changed to “I like orange” and then it will classify the newly created record to understand how “apple” is important. Of course, we need to assume the replaced word (e.g. orange) is noise and it should not provide major change on score.

```
for pipeline in pipelines:
    print('Estimator: %s' % (pipeline['name']))
    labels = pipeline['pipeline'].classes_.tolist()

    if pipeline['name'] in ['Logistic Regression', 'Random Forest']:
        estimator = pipeline['pipeline']
    elif pipeline['name'] == 'XGBoost Classifier':
        estimator = pipeline['pipeline'].steps[1][1].get_booster()
#     Not support Keras
#     elif pipeline['name'] == 'keras':
#         estimator = pipeline['pipeline']
    else:
        continue

    IPython.display.display(
        eli5.show_weights(estimator=estimator, top=10,
        target_names=labels, vec=vec))
```

Estimator: Logistic Regression

y=0 top features		y=1 top features		y=2 top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+2.138	keith	+1.467	graphics	+2.982	windows
+1.141	caltech	+1.040	image	+1.141	file
+1.120	that	+1.014	tiff	+1.098	ini
+0.910	moral	+0.988	images	+0.829	ax
+0.899	is	+0.816	tmc	+0.821	font
+0.895	you	+0.811	file	+0.813	drivers
+0.857	schneider	+0.792	ac	*… 8318 more positive …*	
+0.831	sex	+0.737	linux	*… 23863 more negative …*	
+0.795	livesey	*… 2065 more positive …*		-0.804	in
… 3228 more positive …		*… 30116 more negative …*		-0.886	of
… 28953 more negative …		-0.731	the	-1.387	the
-3.236	<BIAS>	-2.744	<BIAS>	-2.412	<BIAS>

The above figure means that, if input includes “keith”, then score of y=0 increase 2.138. Another case is “the”, it will decrease score -0.731 and -1.387 in y=1 and y=2 respectively.

For local model interpretation, ELI5 use LIME algorithm. The display format is different from LIME but using same idea.

```
number_of_sample = 1
sample_ids = [random.randint(0, len(x_test) -1 ) for p in range(0, number_of_sample)]

for idx in sample_ids:
    print('Index: %d' % (idx))
#     print('Index: %d, Feature: %s' % (idx, x_test[idx]))
    for pipeline in pipelines:
        if pipeline['name'] in ['Logistic Regression', 'Random Forest']:
            estimator = pipeline['pipeline'].steps[1][1]
        elif pipeline['name'] == 'XGBoost Classifier':
            estimator = pipeline['pipeline'].steps[1][1].get_booster()
        #     Not support Keras
#         elif pipeline['name'] == 'Keras':
#             estimator = pipeline['pipeline'].model
        else:
            continue

IPython.display.display(
            eli5.show_prediction(estimator, x_test[idx], top=10, vec=vec, target_names=labels))
```

Contribution?	Feature
+0.078	Highlighted in text (sum)
+0.052	<BIAS>
	… 189 more positive …
	… 27 more negative …
-0.002	if
-0.003	is
-0.004	line
-0.010	writes

from: fwr8bv@fin.af.mil subject: xdm and env. vars organization: the ir
xpert%expo.lcs.mit.edu@fin.lcs.mit.edu hi, i am using xdm on x11r5 wi
used to set path and other environment variables (like manpath, help
therefore neither the olwm root-window nor my applications know abo
/x11/xdm/xdm-config to succesfully pass the path variable. but i am ha

ELI5 explains the test data and deciding that it may be y=0 and the probabilty is 0.031. It also highlight some high positive score and negative score words.

LIME

Photo: https://www.weightlossresources.co.uk/recipes/calorie-counted/drinks/lime-soda-1-pint-309537.htm

As the name mentioned, LIME focus on local model interpretable and model-agnostic part only.

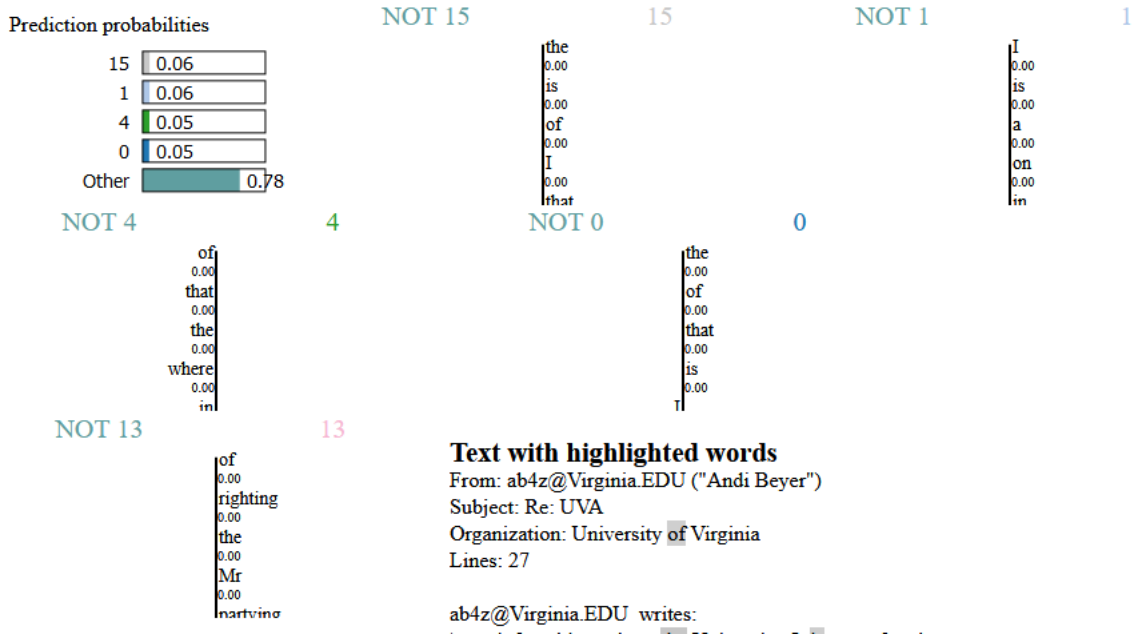
Passing trained model and target record to LIME library. A liner bag-of-words model will be created and providing lots of generated record for training a white box model. The while box model work as a **binary classifier indicating the impact of word existence**. For example, “I like apple”. It will may changed to “I like” and then it will classify the newly created record to understand how “apple” is important.

So no matter what is the provided model is, it can also explain the model by generating records. No matter it is scikit-learn library, XGBoost or even Keras word embedding model. In my sample code, I implemented Keras in sci-kit learn API interface so that I can also use LIME to explain Keras’s word embedding model.

```
number_of_sample = 1
sample_ids = [random.randint(0, len(x_test) -1 ) for p in range(0, number_of_sample)]

for idx in sample_ids:
    print('Index: %d' % (idx))
#     print('Index: %d, Feature: %s' % (idx, x_test[idx]))
    for pipeline in pipelines:
        print('-' * 50)
        print('Estimator: %s' % (pipeline['name']))

        print('True Label: %s, Predicted Label: %s' % (y_test[idx],
pipeline['pipeline'].predict([x_test[idx]])[0]))
```

```
        IPython.display.display(exp.show_in_notebook(text=True))

#         break
#     break
```

```
Estimator: Keras
True Label: 17, Predicted Label: 15
```



LIME builds binary classifiers so we can see that it represent whether it is 0, 9, 15 classes with weight and highlighting high positive and negative score words.

Skater

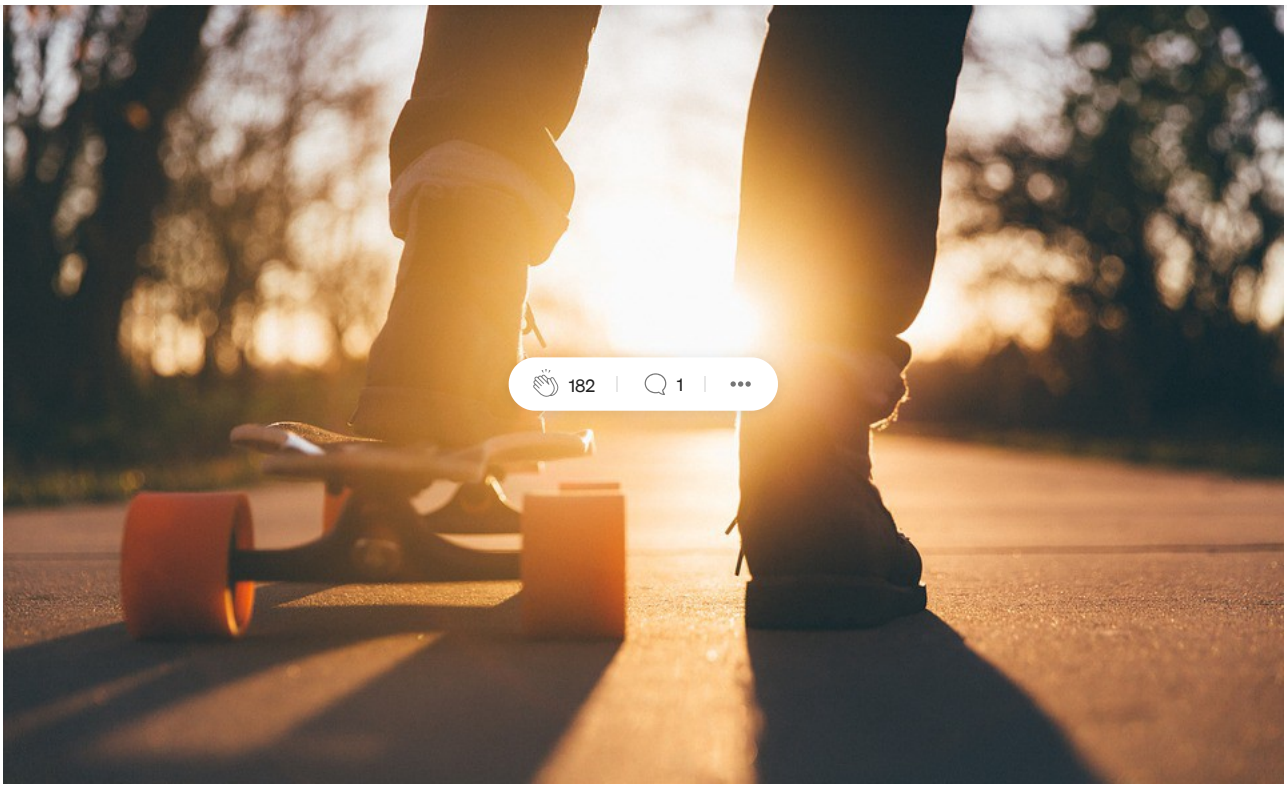


Photo: https://pixabay.com/en/skateboard-youth-skater-boy-skater-1869727/

For global interpretation, Skater provides feature importance and partial dependence approaches.

Tried several time and cannot find a way to provide sci-kit learn pipeline package. Have to convert raw text to numpy format and passing the model separately. Second thing is that it takes about 2 minutes to prepare the feature importance when there is about 800 unique words.

```
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt

from skater.model import InMemoryModel
from skater.core.explanations import Interpretation

transfromed_x_test = vec.transform(x_test[:2]).toarray()

interpreter = Interpretation(transfromed_x_test, feature_names=vec.get_feature_names())

for pipeline in pipelines:
    print('-' * 50)
    print('Estimator: %s' % (pipeline['name']))

    if pipeline['name'] in ['Logistic Regression', 'Random Forest']:
        estimator = pipeline['pipeline'].steps[1][1]
    else:
        continue

    print(estimator)

    pyint_model = InMemoryModel(estimator.predict_proba, examples=transfromed_x_test)

    f, axes = plt.subplots(1, 1, figsize = (26, 18))
    ax = axes
    interpreter.feature_importance.plot_feature_importance(pyint_model, ascending=False, ax=ax)
```

```
                penalty='l2', random_state=None, solver='newton-cg', tol=0.0001,
                verbose=0, warm_start=False)
[159/159] features [                ] Time elapsed: 7 seconds
```

Skater provides the feature importance per word.

Partial dependence [3]is a bunch of figures to represent the relationship (linear, monotonic or complex) between target and features one by one. Providing a visualization, we can visualize the impact between one feature and target. However, we may not use partial dependency if there is NLP.

For local interpretation. Skater wraps LIME module to perform local Text Interpretation. So I will prefer to use native LIME rather than a pure wrapper.

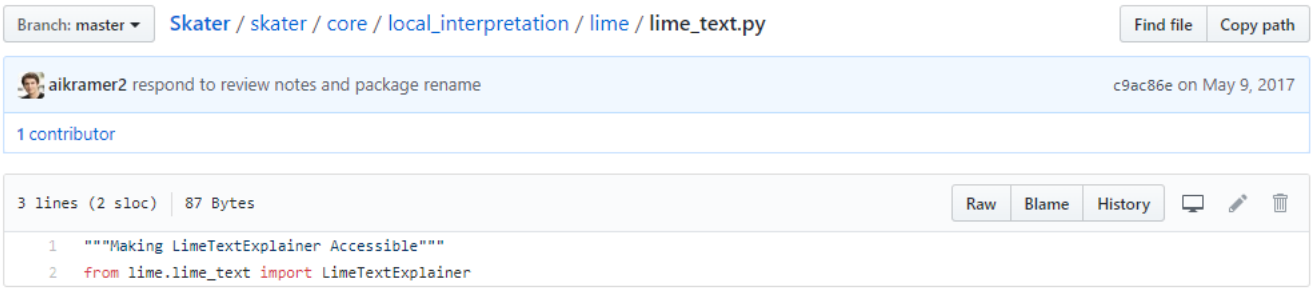


Photo: https://github.com/datascienceinc/Skater/tree/master/skater/core/local_interpretation/lime

Take Away

You can access code via my github repo

- When you read **Christoph's Blog**, you can refer to above code for Local Surrogate Models (LIME) part.
- As a data scientist, we have to explain the model behavior so that **product team and management trust the deliverable** and **debug your model.** Otherwise you may not know what you build.
- For NLP problem, feature importance or permutation feature importance are **demanding as too much features** (word). Not quite useful/ recommend to use it in NLP aspect.
- LIME **do not able to explain the model correctly** on some scenario. For example, the score can be different every time as it generates samples randomly. Will have another article to introduce how to resolve it.
- For ELI5, **sci-kit learn pipeline** is supported in show_weight (Global Interpretation) but it is **not supported in show_prediction** (Local Interpretation) **in ELI5.** Reason is mentioned in here. To overcome it, you can simply use LIME directly.
- There is bug when using ELI5 (version: 0.8) if the classifier is XGBoost Classifier. If you use **XGBoost classifier, have to perform some workaround** due to ELI5 bug (xgb_classifier.get_booster())
- For Keras model, implemented sci-kit learn API interface but still cannot use ELI5. ELI5 will check the object type to prevent unexpected library.

Visit my blog from http://medium.com/@makcedward/

Get connection from https://www.linkedin.com/in/edwardma1026

Explore my code from https://github.com/makcedward

Check my kernal from https://www.kaggle.com/makcedward

Reference

- [1] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. "Why Should I Trust You?" Explaining the Predictions of Any Classifier. Sep 2016. https://arxiv.org/pdf/1602.04938.pdf
- [2] Breiman Leo. " Random Forests".Jan 2001. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
- [3] Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." Apr 2001. https://statweb.stanford.edu/~jhf/ftp/trebst.pdf