About Me.

Name	:	Ashwin Bhat
University	:	Indian Institute of Information Technology, Kurnool, INDIA. (IIITK)
Contact Info	:	Email : ashwinbhat2906@gmail.com
		Phone : +91 8754532125
Time Zone	:	+5:30 GMT
Resume Link	:	AshwinB Resume

Code Contribution.

- Catboost Support : In this pull request, I have integrated the catboost support for eli5's explain_weights feature. The code was developed with continuous testing and all modules were tested before being commited. I had assistance from one of the project mentors Konstantin Lopuhin.

Project Information.

Sub-org Name:

Scrapy ELI5

Project Abstract:

Model explainability is a priority in today's machine learning community. ELI5 is a library which provides a **uniform** api for explaining different kind of models and their predictions. This project is intended towards adding state of art model explainability libraries to ELI5.

Detailed Description:

SHAP(2017) and LIME(2016) are both popular Python libraries for model explainability. SHAP (SHapley Additive exPlanation) leverages the idea of Shapley values for model feature influence scoring. LIME (Local Interpretable Model-agnostic Explanations) builds sparse linear models around each prediction to explain how the black box model works in that local vicinity.

SHAP paper was published one year after the LIME paper. And upon further inspection it can be concluded that SHAP is the superset of LIME.

SHAP vs LIME

- LIME is fast, while Shapley values take a long time to compute
- the Authors of SHAP show that Shapley values provide the only guarantee of accuracy and consistency and that LIME is actually a subset of SHAP, and lacks the same properties.
- The SHAP Python library helps with this compute problem by using approximations and optimizations to greatly speed things up while seeking to keep the nice Shapley properties.
- SHAP is not optimized for all model types yet. Hence some LIME interpretations for models can be faster than SHAP.

SHAP and LIME implementations are separate projects in themselves. I have incorporated the whole timeline of integration of SHAP features and tests. If I am able to stay ahead of the timeline I also propose to add LIME for unsupervised models.

SHAP implementation

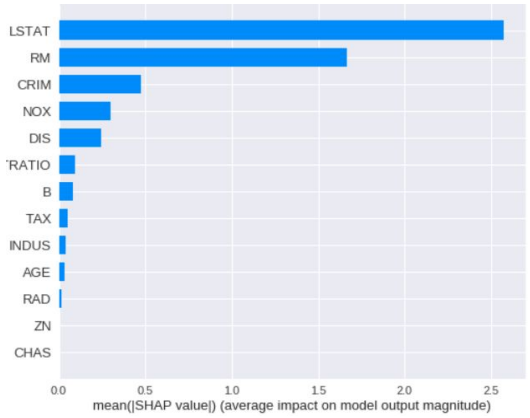
The SHAP library has three core Implementation namely **TreeExplainer, KernelExplainer, DeepExplainer(**won't be used**).**

1. **TreeExplainer :** Tree SHAP is a fast and exact method to estimate SHAP values for tree models and ensembles of trees, under several different possible assumptions about feature dependence. It depends on fast C++ implementations either inside an external model package or in the local compiled C extension.
2. **KernelExplainer:** Kernel SHAP is a method that uses a special weighted linear regression to compute the importance of each feature. The computed importance values are Shapley values from game theory and also coefficients from a local linear regression.

Overview of ELI5

EXPLAIN_WEIGHTS

ELI5 has Explanation Class to explain the feature importance w.r.t to gain,mean error to explain the feature importance.



Weight	Feature
0.4278	Sex=female
0.1949	Pclass=3
0.0665	Embarked=S
0.0510	Pclass=2
0.0420	SibSp
0.0417	Cabin=
0.0385	Embarked=C
0.0358	Ticket=1601
0.0331	Age
0.0323	Fare
0.0220	Pclass=1
0.0143	Parch
0	Name=Rothes, the Countess. of (Lucy Noel Martha Dyer-Edwards)
0	Name=Roebling, Mr. Washington Augustus II
0	Name=Rosblom, Mr. Viktor Richard
0	Name=Ross, Mr. John Hugo
0	Name=Rush, Mr. Alfred George John
0	Name=Rouse, Mr. Richard Henry
0	Name=Ryerson, Miss. Emily Borie
0	Name=Ryerson, Miss. Susan Parker "Suzette"

This Same Explanation class can be recreated for Mean[SHAP] importance by generating the shap values and displaying them in a similar format by using the Explanation Module from **eli5 base**

.

The details how these changes will be implemented are mentioned in the IMPLEMENTATION section.

EXPLAIN_PREDICTION

This Api is used to give the feature contribution towards each feature given a data point.

y=1 (probability **0.566**, score **0.264**) top features

Contribution[?]	Feature	Value
+1.673	Sex=female	1.000
+0.479	Embarked=S	Missing
+0.070	Fare	7.879
-0.004	Cabin=	1.000
-0.006	Parch	0.000
-0.009	Pclass=2	Missing
-0.009	Ticket=1601	Missing
-0.012	Embarked=C	Missing
-0.071	SibSp	0.000
-0.073	Pclass=1	Missing
-0.147	Age	19.000
-0.528	<BIAS>	1.000
-1.100	Pclass=3	1.000

For Tree and Kernel Based models the Shapley Values extracted will be used to predict the explainability of trained weights. For Black Box models, a shapley value based linear Regressor will be used to predict the explainability of the trained weights.

Reasons for not wrapping the existing SHAP library.

The shap library will include a number of extra dependencies for **eli5**, namely
• C extensions,
• `install_requires=['numpy', 'scipy', 'scikit-learn', 'matplotlib', 'pandas', 'tqdm', 'ipython', 'scikit-image']`
• no Linux wheels (so installation on Linux can be not straightforward),
• unclear supported Python versions (it looks like Python 3.6 and 3.7 are supported - https://pypi.org/project/shap/#files, but they run tests only on Python 3.6, and wheels are not available for all combinations of platform + Python, e.g. Python 3.7+OSX == no luck getting a wheel; we support more than Python 3.6 and 3.7 in eli5.

Implementation.

After discussion with the mentors the Implementation was narrowed down into Three Phases.

1. Native SHAP from tree models.
2. SHAP for the rest of tree models
3. Black-box models

Native SHAP from tree models providing it

XGBOOST, LIGHTGBM, CATBOOST have optimised C implementations for computing shapley values. A function **get_feature_importance(`model`,type='shap').** The process of extraction will include type checking on data types and model types and then calling their functions to get the values which will be formatted accordingly and based to the **eli5.base.Explanation** to display.

Native SHAP for sklearn tree Based Models

Random Forest Regressor, Extra Trees Regressor, Decision Trees Regressor, Decision Trees Classifier, Random Forest Classifier, ExtraTrees Classifier, GradientBoostingRegressor,Gradient

Boosting Classifier are Tree Based Models provided by the Sklearn Api. These models do not provide a native SHAP implementation to generate shapley importance values. I propose Implementing native SHAP calculator in python by Dissecting the TreeExplainer of the SHAP library.

Black-box models

For Black Box models the decided implementation for explaining weights discussed was by having NATIVE implementation of the Kernel Explainer that makes use of a linear regressor of sorts to fit/calculate the shapley values and then uses the model to explain the predictability of the model. The actual implementation is subject to discussion w.r.t to the KernelExplainer from the SHAP library serving as reference.

Ranking from the Top in terms of increasing difficulty, which makes the Black Box Model the most challenging and hence will be attacked after the first month and a half of coding if the previous implementations get approved.

API Structure

Eli5 →shap →Tree →XGBOOST
→CATBOOST
→LIGHTGBM
→Scikit Learn →Random Forest Regressor
→Extra Trees Regressor
→Decision Trees Regressor
→Decision Trees Classifier
→Random Forest Classifier
→Extra Trees Classifier
→GradientBoostingRegressor
→Gradient Boosting Classifier
→XGBoost Classifier/Regressor
→LightGBM Classifier/Regressor

→Explain →explain_weights

→explain_prediction

Proposed Timeline

Community Bonding(May 7-26): I wish to explore further into the explain prediction module with help from mentors and get clear on that aspect. Also discuss any further application of eli5 module using SHAP interaction values.

Week 1 (May 27-31): Integration of Explainers by using the native shap implementations of r CATBOOST,XGBOOST and LIGHTGBM.

Week 2 (June 3): Testing(unit tests) the Explainers of the above classifiers.

Week 3 (June 10): Explain_weights integration for multiple scikit based Xgboost, LightGBM Tree based Classifiers/Regressors

Week 4 (June 17): Extensive Testing of all the integration for Explain_weights.SHAP

Week 5 (June 24): Explain_weights integration for multiple Tree based Classifiers, namely, Random Forest Regressor, Extra Trees Regressor, Decision Trees Regressor, Decision Trees Classifier, Random Forest Classifier, ExtraTrees Classifier, GradientBoostingRegressor,Gradient Boosting Classifier.

Week 6 (July 1): Explain_weights integration for multiple Tree based Classifiers, namely, Random Forest Regressor, Extra Trees Regressor, Decision Trees Regressor, Decision Trees Classifier, Random Forest Classifier, ExtraTrees Classifier, GradientBoostingRegressor,Gradient Boosting Classifier.

Week 7 (July 8): Explain_weights integration for multiple Tree based Classifiers, namely, Random Forest Regressor, Extra Trees Regressor, Decision Trees Regressor, Decision Trees Classifier, Random Forest Classifier, ExtraTrees Classifier, GradientBoostingRegressor,Gradient Boosting Classifier.

Week 8 (July 15): Testing/Working on Tree Based Classifiers. and black box models.

Week 9 (July 22): If timeline is adhered to. Implementing the black box models for explain_weights .. And extending API.

Week 10 (July 29): Testing and documentation. Will be done on the go, hence extra time if any can be utilised to add multiple example support to explain_predicitons.

Week 11 (August 5): Code Freeze and updating leftover documentation

Week 12 (August 12): Final checks to make the PR successful.

Final week (August 19): This week you will be submitting your projects

I am used to Test Driven Development. So as the development progresses I will be writing unit tests side by side. The Testing periods are specifically mentioned to incorporate any edge cases or scenarios that haven't occurred to me during coding.

Other Commitments.

- My college ends April 28th. I don't have classes but a Final Exam. So I might not be very responsive the last week of April.
- Other than that, I am free throughout this summer which I intend to make good use of.

Thank you!
It will be an honour to work with you and learn/grow along the process.

REFERENCES

1. "Consistent Individualized Feature Attribution for Tree Ensembles." 12 Feb. 2018, https://arxiv.org/abs/1802.03888.
2. "A Unified Approach to Interpreting Model Predictions." 22 May. 2017, https://arxiv.org/abs/1705.07874.
3. "GitHub - slundberg/shap: A unified approach to explain the output of ...." https://github.com/slundberg/shap. Accessed 6 Apr. 2019.