EFFCT PLOT AND PCA STRENGTHENING

Personal Details:

- Name: Naresh Bachwani
- Email: nbnb76543@gmail.com
- GitHub link: https://github.com/naresh-bachwani
- LinkedIn: https://www.linkedin.com/in/naresh-bachwani/
- Phone No: +91-8112232962,

Project Details:

Abstract:

YellowBricks is an open source python data visualisation library aiding both exploratory data analysis and machine learning tasks. As my Google Summer of Code project, I would like to propose building a new visualizer “Effect Plot” to help in interpreting linear model. Along with effect plots, secondary aim for the project would be to extend PCA visualizer by adding features like optional heatmap, colorbar and alpha params. Effect plot tells the user about the effect various features of a dataset would have when a dataset is trained upon a particular linear model. The effect plot is basically a boxplot explaining the variance and medians of the effect of each feature. The aim of this project is to cater to need of user by providing them control over various aspects of effect plots like face color, line color, linewidth, and shape and color of outliers along with many others hyperparameter to tune.

Challenges:

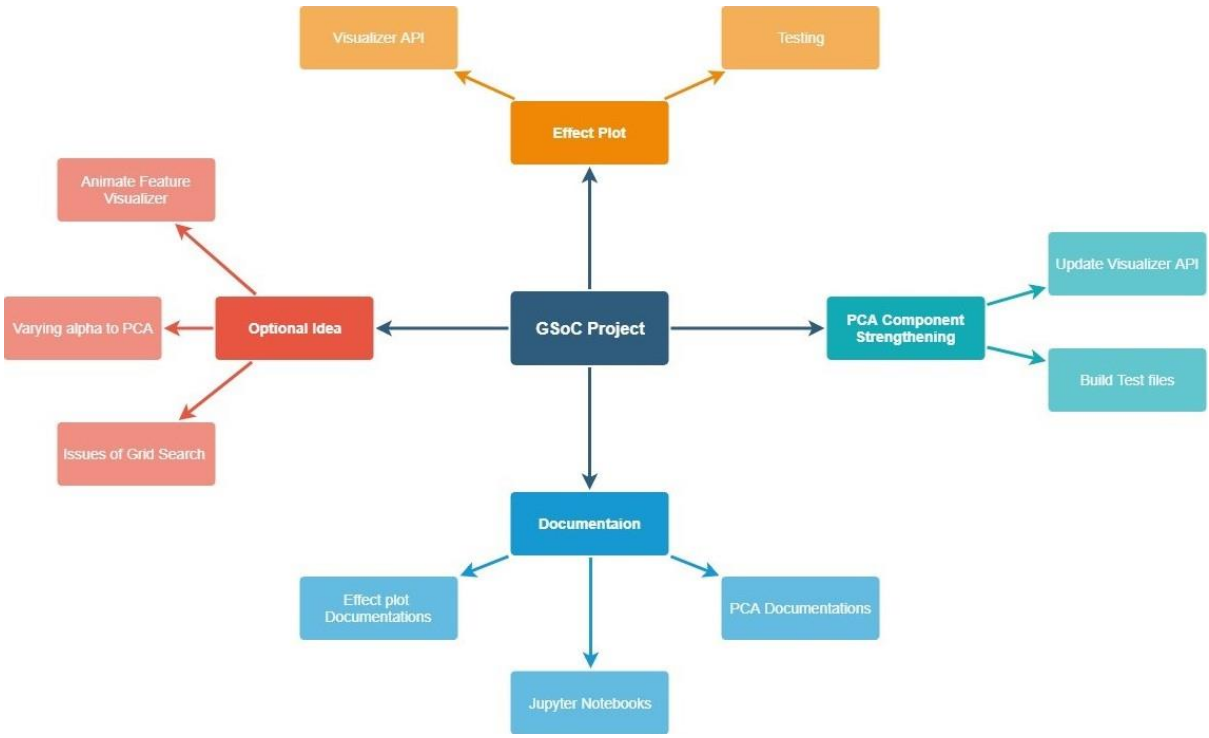
Yellowbricks already has a visualiser for feature importance of classifier but no feature importance for regressors. The challenge is to make a effect plot visualizer for linear models and fine-tune the existing PCA

visualizer. The question while designing a visualizer is “who are the users?”. The yellowbrick “users” are data analyst who aim to produce plots with varied flavours based on their needs. To capture this spectrum, a thorough study of a visualiser and multiple repetitions of experiments are required.

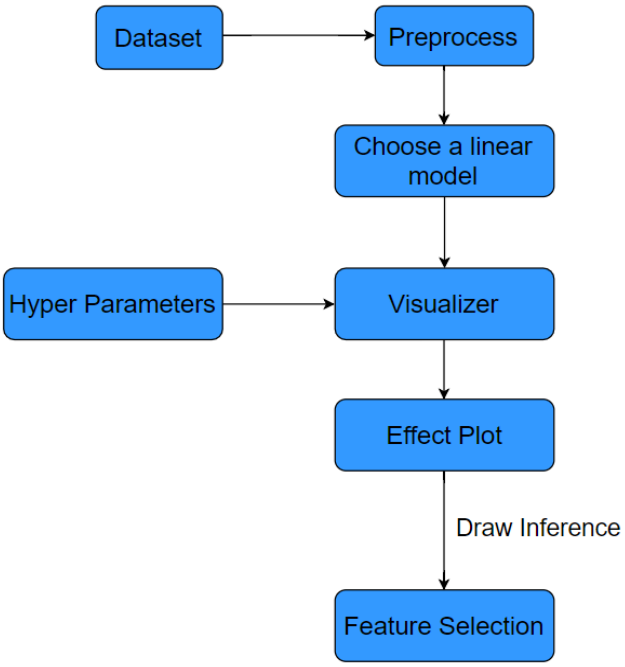
I feel having a well-built structured strategy combined with the target-oriented effort on YellowBricks library will be more than half the work done.

Technical Details:

The GSoC project is divided into following major categories with effect plot being the primary focus followed by PCA component strengthening.



The basic pipeline for dealing with effect plot would look like:

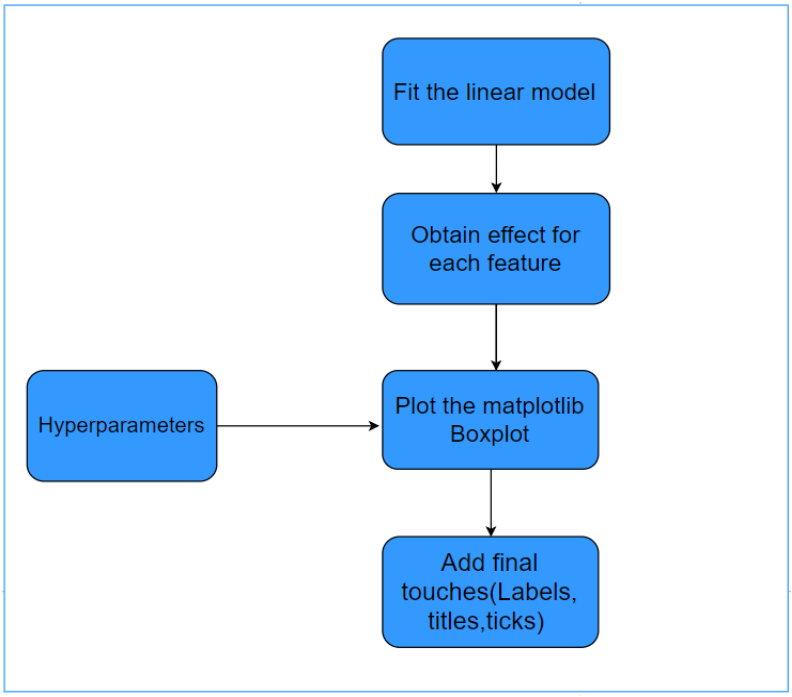


Building a visualizer can be divided into three components. Visualizer API – Testing – Documentation.

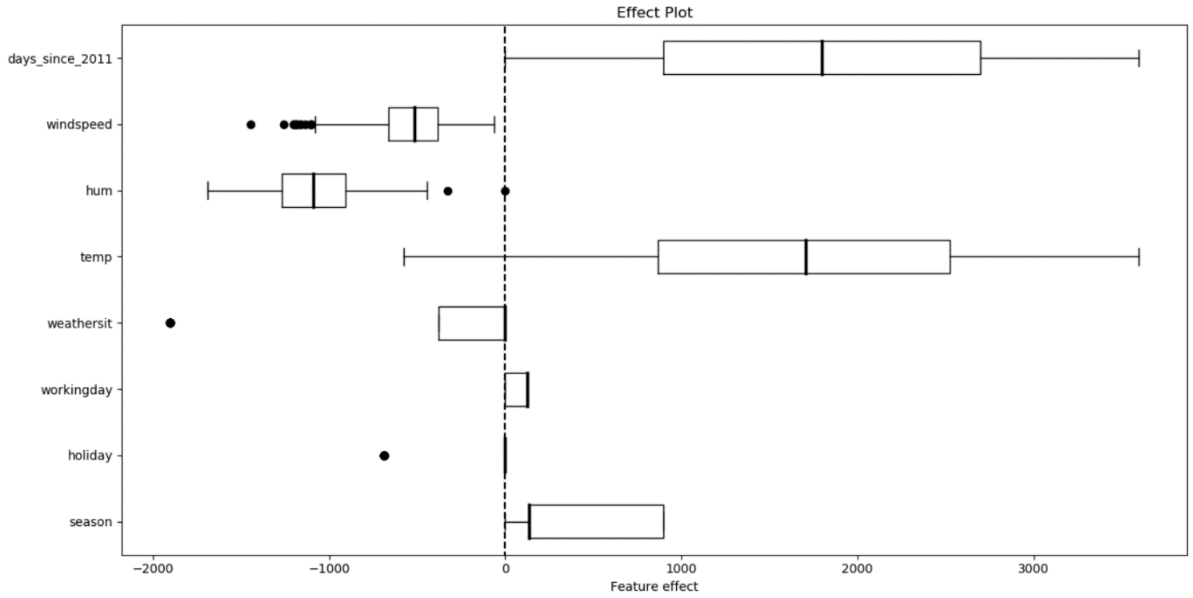
Visualizer API	Testing	Documentation
• Baseline model • Adding hyperparameters • Testing on all linear models • Structure to suit YellowBricks • Adding relevant functionality • Generate more better results	• Add visualizer and data mix-in test. • Build Integration test • Asserting that various	• Docstring in required format • Adding sample codes

Part 1: Visualizer API

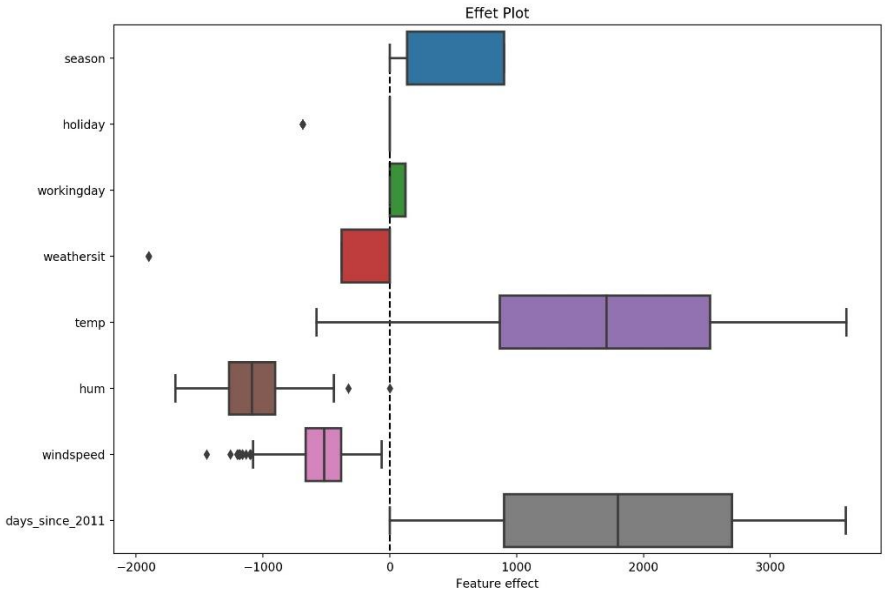
As pointed out earlier, the effect plot needs to provide users with considerable amount of control over the what the plot would look like. I experimented to make one of the initial plots by building a class showing only basic effect plot with almost no hyperparameters. Since then I made quite a lot of changes by working around with various related visualizers and gradually learning about the structure for building a class based on YellowBrick format. The pipeline for the effect plot visualizer would look like this



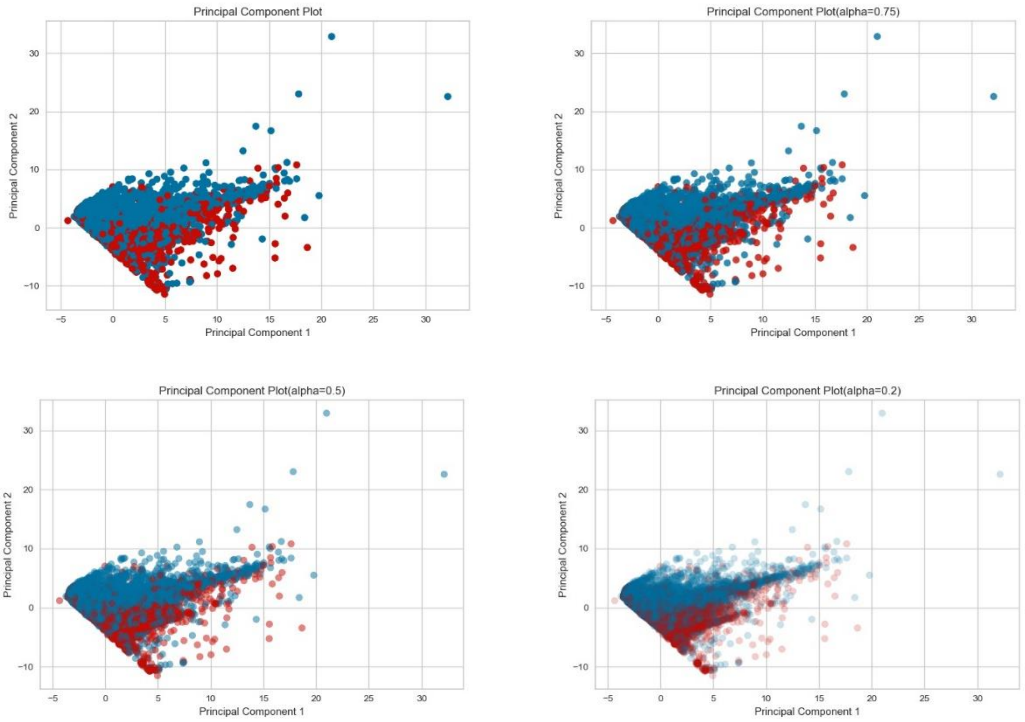
The very first plot I made without knowing much



The plot to be obtained after working with the various visualizer and getting familiar with their structures.

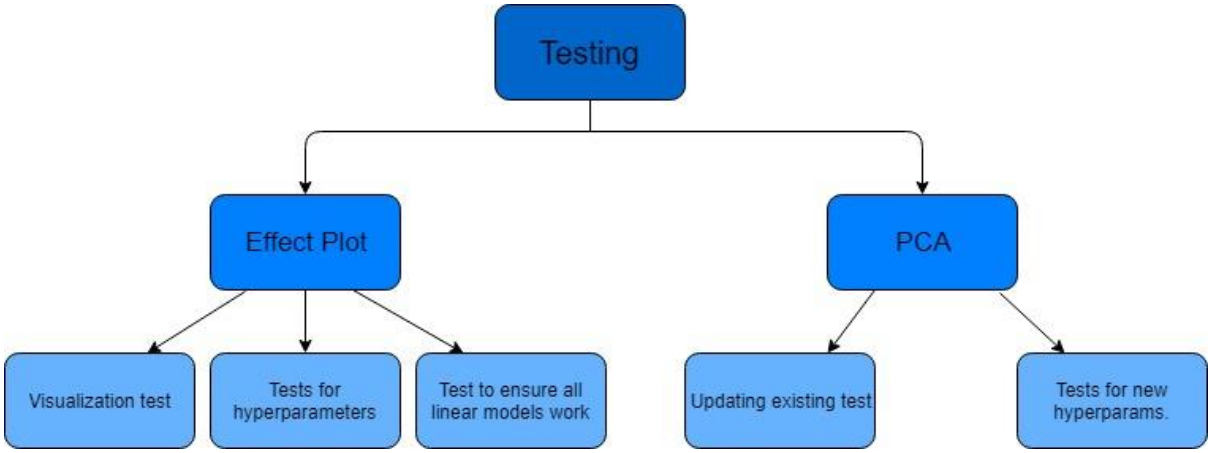


The hyperparameters gives the users control over the finalized plot. Adding hyperparameter like optional colorbar and heatmap would mark an important milestone for PCA visualizer. Other parameters like alpha and random state also need to be dealt with. Adding alpha param would make the PCA visualizer look less crowded. A sample class built to deal to add alpha params gave following results.



Part 2: Testing

Testing a visualizer is usually done to make sure that its output matches the pre-defined notions of YellowBricks. I plan on making a test file through periodic consultation with the mentors so that it agrees with the structure of YellowBricks. The workflow for testing would be like this



Part 3: Documentation

Documentation is an important aspect for successful deployment of a visualiser. The initial step would be to add a well structured docstring to the visualiser API written in NumPy format. In the later stages I plan on adding sufficient amount of example code to cover almost every aspect of using the effect plot visualiser. I also plan to update the existing PCA documentation.

Optional Ideas

- Adding an optional feature to plot data points distribution over boxplot.
- Add varying alpha for different classes for discrete data distribution in PCA
- Work to resolve few issues of Grid Search expansion (The coding part)
- Work on improving the PCA pipeline.

Proposed Timeline

Deciding timeline for a project is quite an effort. I will lay out my project goals for this summer.

Community Bonding

Spend this time learning more about dependencies of Yellowbricks, Matplotlib and Scikit-Learn. Most of the time would be devoted towards deciding the structure of the class by regular discussions with the mentors. Clear idea of structure is necessary so as to avoid bad decisions early on in project. I would also like to explore and exploit any idea that can help in making the visualiser better. A deliverable here would be an exact spec of what code needs to be written.

Phase 1(May 27 – June 28)

Time to start coding! This phase will be divided in two parts:

1. **Visualizer APIs:** The plan of attack here will be to code basic visualizer and then adding hyperparameters to the class. The code snippet for fit() would look like:

```
self.estimator.fit(X,y)
coeffs = self.estimator.coef_[0]
effect=pd.DataFrame()
for cols in unique_col:
    a = [cols in a for a in newdf1.columns.values]
    col_names = list(newdf1.columns.values[a])
    col_coeffs = coeffs[a]
    effect[cols] = (newdf1[col_names]*col_coeffs).sum(axis=1)
a = [a in non_unique for a in newdf1.columns.values]
col_coeffs = coeffs[a]
effect[non_unique] = (newdf1[non_unique]*col_coeffs)
self.effect = effect
self.draw(effect)
```

2. **Test files:** Adding various tests by coding various small tests to build a test class.

Phase 2(June 29 – July 26)

This phase will deal with the PCA visualizers. The major steps for finetuning visualizer API would include:

1. **Hyperparameters:** This will involve writing code, which adds hyperparameters to class.
2. **Testing:** This will involve updating various tests. The plan would be to update the visualizer tests and adding various other tests do deal with adding new parameters.

Phase 3(July 27 – August 19)

This phase would involve clean-up of the code written in the earlier phases, testing, bug fixes and ensuring that the features added during the summer is usable. Another goal would be to come with ample number of examples and adding Jupyter notebooks to make the visualizer easy to start with.

After Summer of Code: I plan to continue working on our efforts of YellowBricks and ensure that all issues regarding effect plot and PCA visualizer are being dealt with. I plan on working with new visualizer like adding animation visualizer for feature visualizer.

A detailed week by week timeline for 12 weeks period of GSoC coding period to keep me on organised:

- **Week 1:** Begin writing the class for Effect Plots. Make relevant pull requests and test basic effect plot model.
- **Week 2:** Adding various hyperparameters while testing against the various possible bugs.
- **Week 3:** Adding basic visualization test and scaling up to integration test for linear variables.
- **Week 4:** If lagging on work, then cover up. If not, then continue with plan, with an intent to get some optional ideas started in the future. Also prepare for phase 1 evaluation
- **Week 5:** Add code to strengthen the PCA. Adding heatmap and colorbar will be primary focus.
- **Week 6:** Finish the task from week 5 and add other parameters to the class.
- **Week 7:** Update the test files for PCA and add more tests. Also try to remove the existing bugs.

- **Week 8:** Taking mentor evaluation into account, fix stuff in code written so far and continue coding. And also prepare the phase 2 evaluation.
- **Week 9:** Begin on documentation part and explore the possible application of the visualizers to cover all examples
- **Week 10:** If lagging on stuff, then cover up. If not, then work on optional ideas.
- **Week 11:** Prepare the Jupyter notebooks. Also test for possible errors i.e. try to make code bug free.
- **Week 12:** Work towards final submission and make sure that everything is okay.

Why this Project?

- The proposed project would help me develop and refine my development skills.
- I get to learn about data visualization which will help in my career goal to become a data analyst.
- Yellowbricks provide visual diagnostics of the models and visual steering which prove quite useful in exploratory data analysis required for a successful model.

Development Experience

I began coding in the senior year of my high school, building small Java applications. I shifted to python in my sophomore year in graduation and participated in competitive programming. I began learning machine learning by the end of my second year and have been developing small project since then. I have worked with data visualization as a crucial part for some project. Few of my projects include constructing an interactive application for displaying our college results and Fake Image Recognizer.

Detailed Information about yourself:

I am a junior undergrad at National Institute of Technology, Hamirpur. My major is in Electronics and Communication. I came across YellowBricks when looking alternative libraries for seaborn. I spent few weeks to understand the composition of library and learn about its working. The community and the work made me fall in love with them. The enthralling work is the reason I'm applying for GSoC. Currently I do not have an active blog but will be blogging about the progress of my project once it gets started. My activities on yellowbricks so far is as follows:

- I did some work on issue to animate the feature visualizer.
- I have made following pull requests:
    1. Added alpha param to PCA.
    2. Added train and test alpha to residuals.
- I also raised few issue and interacted with community members.
- I also designed a base class for effect plots and have submitted my results here.