Final Presentation

Betas
Visualization

Arjun Singh, Cathy Jia, Joel Stremmel, Monique Bi, Yiming Liu

Project Background

- A simple visualization tool
- A pip installable library
- Designed for data scientists or analysts who:
    - Do not want to generate custom plots for analyzing model scores or model assumptions
    - Use ML models without a detailed understanding of how the models work and would like to rely on a pre-configured solution for applying commonly used methods
    - Are familiar with: Python and command line

Data Sources

➔ *betas* makes it easy to analyze scores from models trained on a variety of data sources

		Spam Data	Breast Cancer Data	Auto Data	College Data
Number of Instances		4601	569	398	777
Number of attributes	Continuous	57	30	5	19
	Discrete / Nominal	1	1	4	2
Source		ESL*	scikit-learn	ISL*	ISL*

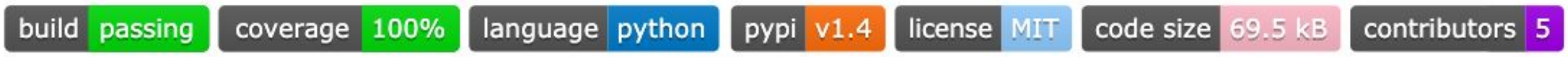
* ESL: *The Elements of Statistical Learning*; ISL: *An Introduction to Statistical Learning*

Software & License

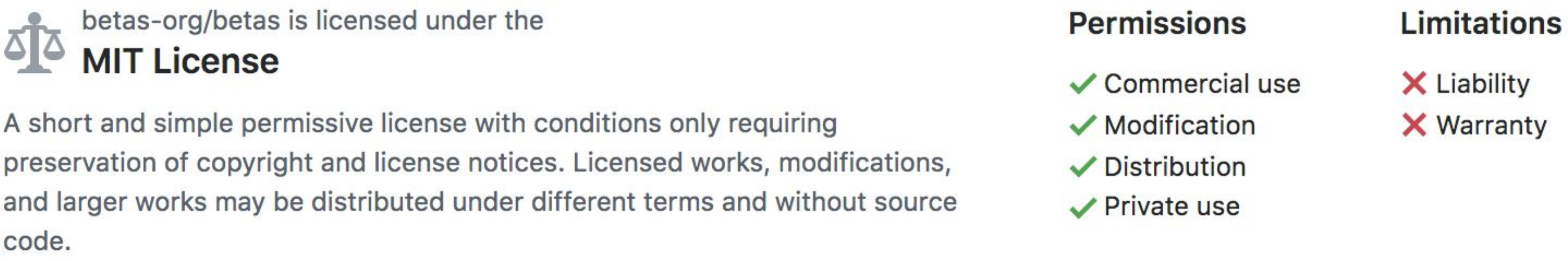
```
pip install betas
```

❑ **Python 3.6 / 3.7**

❑ Passing build with 100% test coverage



build passing | coverage 100% | language python | pypi v1.4 | license MIT | code size 69.5 kB | contributors 5



betas-org/betas is licensed under the

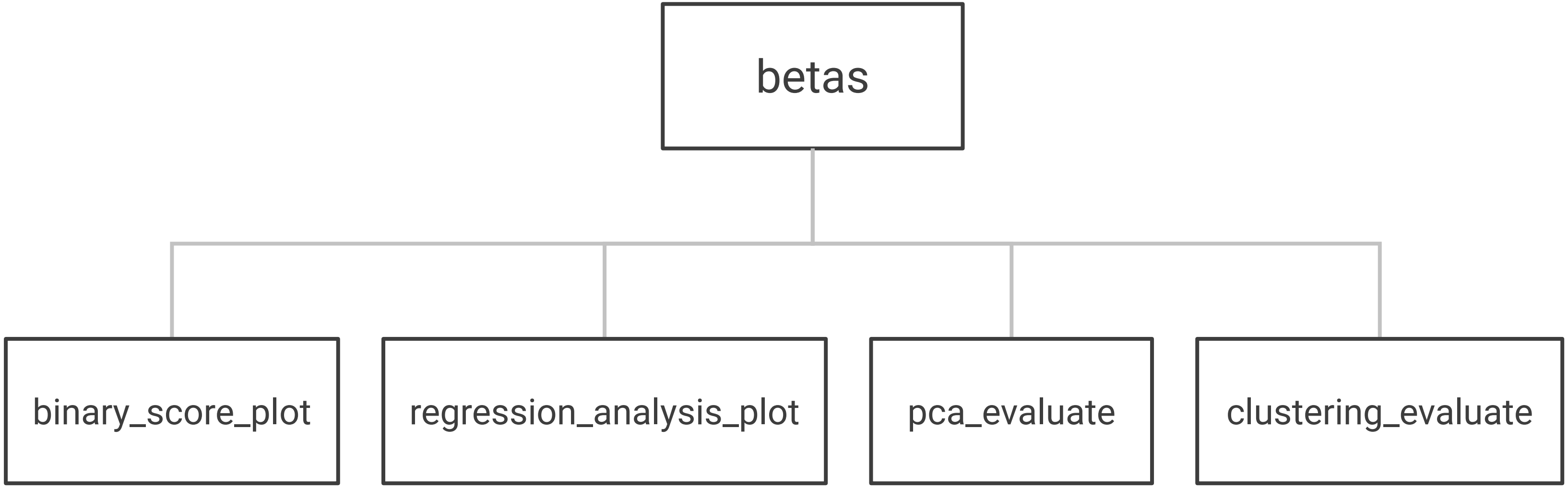
MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Limitations
✓ Commercial use	✗ Liability
✓ Modification	✗ Warranty
✓ Distribution	
✓ Private use	

betas Library

Individual modules support evaluating four key data science methods



betas

binary_score_plot

regression_analysis_plot

pca_evaluate

clustering_evaluate

betas Library

		Unit Test	Documentation	Related demo files
1	binary_score_plot.py	✓	✓	• test_binary_score_plot.py • demo_binary_scoe_plot.ipynb • binary_score_diagnostics.py
2	regression_analysis_plot.py	✓	✓	• test_regression_analysis_plot.py • demo_binary_scoe_plot.ipynb • binary_score_diagnostics.py
3	pca_evaluate	✓	✓	• test_pca_evaluate.py • demo_pca_evaluate.ipynb
4	clustering_evaluate	✓	✓	• test_clustering_evaluate.py • demo_clustering_evaluate.ipynb

Use Cases

Four use cases:

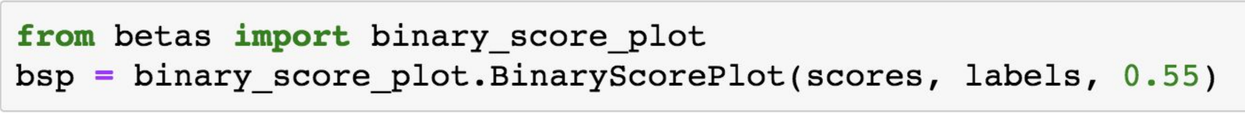
1. Fit binary classification model
2. Fit linear regression model
3. Evaluate PCA performance
4. Evaluate clustering performance

With betas, users can

- ★ Install and import betas
- ★ Input an interesting dataset
- ★ Run simple code
- ★ Create plots and explore dataset

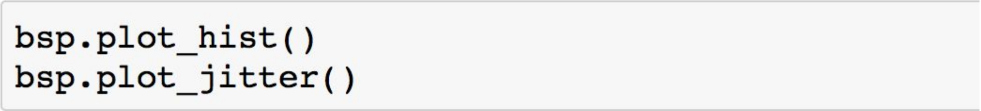
Binary Classification: Binary Score Plot

- Histogram of model scores
- Scatterplot of model scores vs. actual labels
- ROC curve
- Precision-recall curve
- Optimal threshold

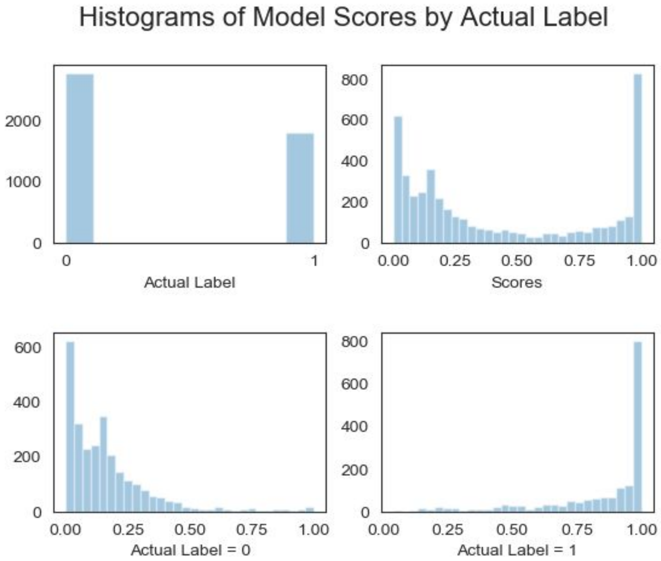


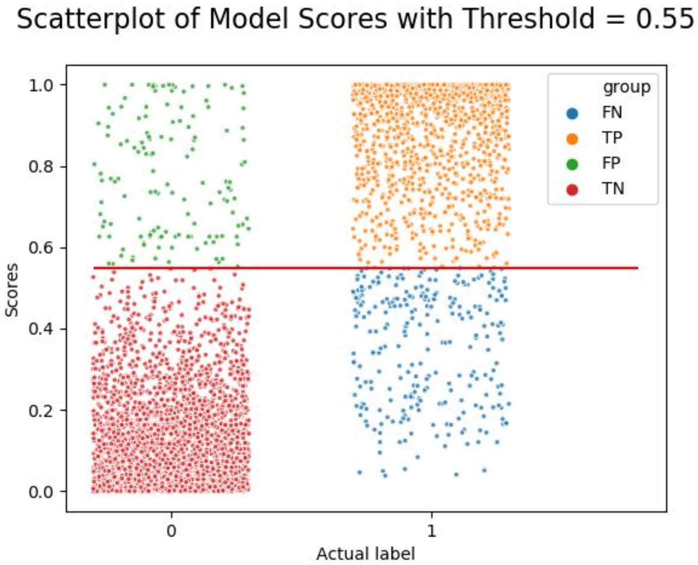
```
from betas import binary_score_plot
bsp = binary_score_plot.BinaryScorePlot(scores, labels, 0.55)
```

Binary Classification: Binary Score Plot

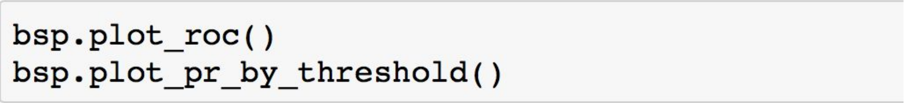


```
bsp.plot_hist()
bsp.plot_jitter()
```

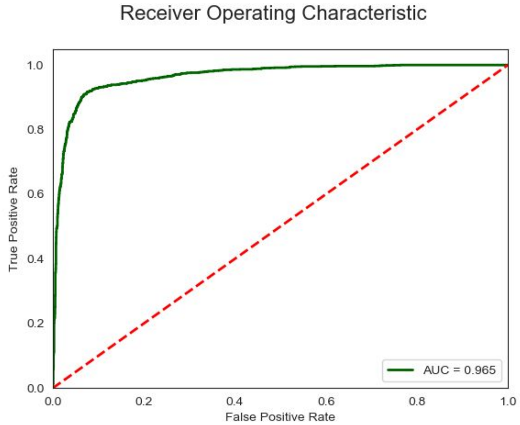

Histograms of Model Scores by Actual Label

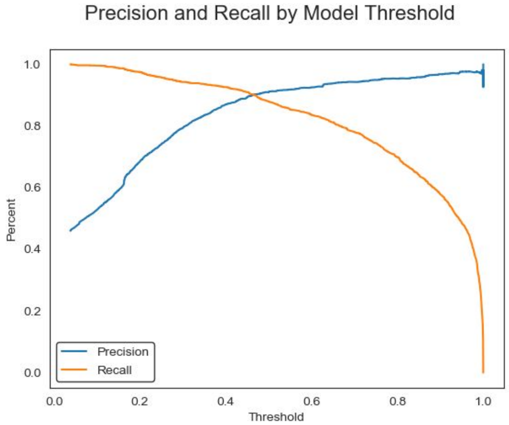

Scatterplot of Model Scores with Threshold = 0.55

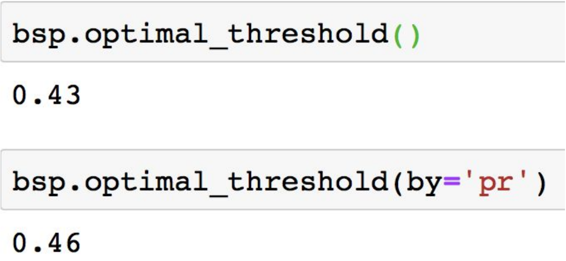
Binary Classification: Binary Score Plot



```
bsp.plot_roc()
bsp.plot_pr_by_threshold()
```







```
bsp.optimal_threshold()
```

```
0.43
```

```
bsp.optimal_threshold(by='pr')
```

```
0.46
```

Binary Classification: Dashboard

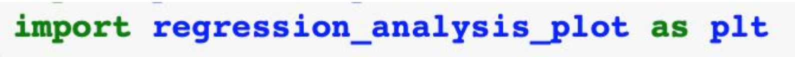
- Interactive dashboard built with bokeh
- Helps users better understand the distribution of modeled scores
- Demo

Linear Regression: Analysis Plot

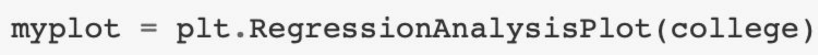
User Input

- A dataframe
- A list of predictor variable(s)
  - *optional**
- A response variable
  - *optional**

** Predictor(s) and response can be reselect*



```
import regression_analysis_plot as plt
```



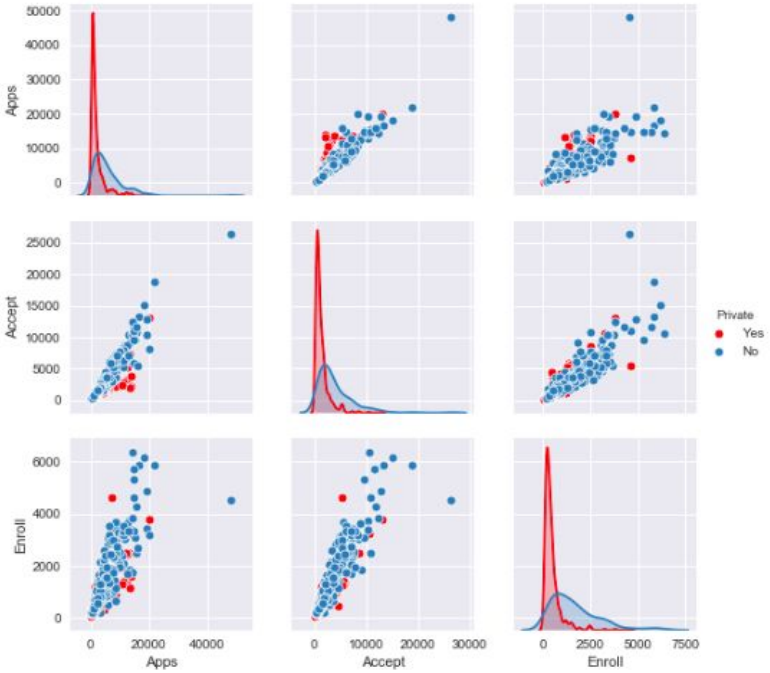
```
myplot = plt.RegressionAnalysisPlot(college)
```

Main Methods

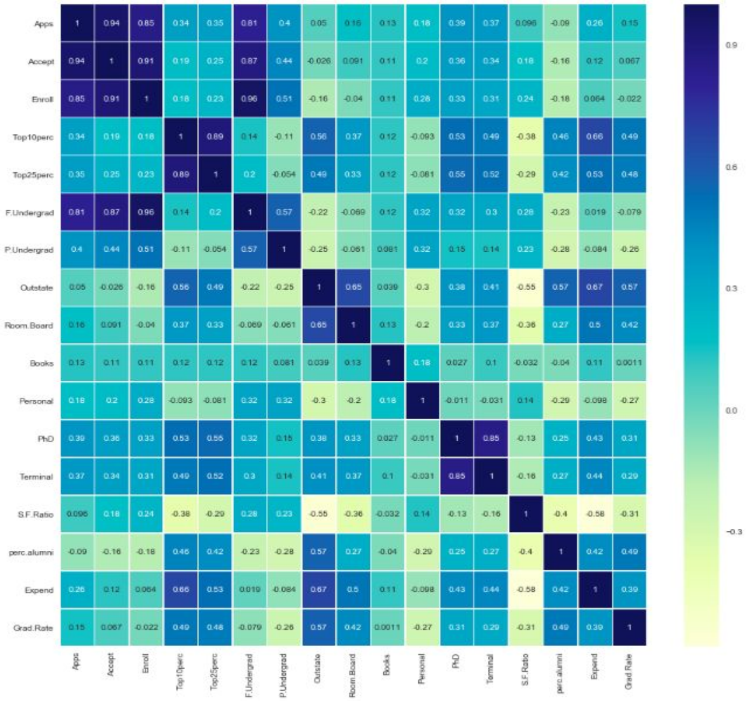
- Scatter matrix plot
- Correlation heatmap
- Box plot
- Distribution plot
- Scatter plot with regression line
- Basic linear regression model
- Model diagnostics

Linear Regression: Analysis Plot Demo

```
myplot.matrix_plot()
```

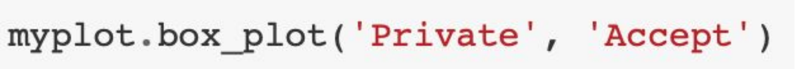


```
myplot.corr_heatmap(figsize=(15,15))
```

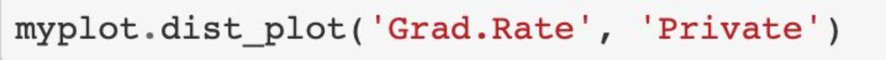


	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
Apps	1	0.94	0.85	0.34	0.35	0.81	0.4	0.05	0.16	0.13	0.18	0.39	0.37	0.096	-0.09	0.26	0.15
Accept	0.94	1	0.91	0.19	0.25	0.87	0.44	-0.026	0.091	0.11	0.2	0.36	0.34	0.18	-0.16	0.12	0.067
Enroll	0.85	0.91	1	0.18	0.23	0.96	0.51	-0.16	-0.04	0.11	0.28	0.33	0.31	0.24	-0.18	0.064	-0.022
Top10perc	0.34	0.19	0.18	1	0.89	0.14	-0.11	0.56	0.37	0.12	-0.093	0.53	0.49	-0.38	0.46	0.66	0.49
Top25perc	0.35	0.25	0.23	0.89	1	0.2	-0.054	0.49	0.33	0.12	-0.081	0.55	0.52	-0.29	0.42	0.53	0.48
F.Undergrad	0.81	0.87	0.96	0.14	0.2	1	0.57	-0.22	-0.069	0.12	0.32	0.32	0.3	0.28	-0.23	0.019	-0.079
P.Undergrad	0.4	0.44	0.51	-0.11	-0.054	0.57	1	-0.25	-0.061	0.081	0.32	0.15	0.14	0.23	-0.28	-0.084	-0.26
Outstate	0.05	-0.026	-0.16	0.56	0.49	-0.22	-0.25	1	0.65	0.039	-0.3	0.38	0.41	-0.55	0.57	0.67	0.57
Room.Board	0.16	0.091	-0.04	0.37	0.33	-0.069	-0.061	0.65	1	0.13	-0.2	0.33	0.37	-0.36	0.27	0.5	0.42
Books	0.13	0.11	0.11	0.12	0.12	0.12	0.081	0.039	0.13	1	0.18	0.027	0.1	-0.032	-0.04	0.11	0.0011
Personal	0.18	0.2	0.28	-0.093	-0.081	0.32	0.32	-0.3	-0.2	0.18	1	-0.011	-0.031	0.14	-0.29	-0.098	-0.27
PhD	0.39	0.36	0.33	0.53	0.55	0.32	0.15	0.38	0.33	0.027	-0.011	1	0.85	-0.13	0.25	0.43	0.31
Terminal	0.37	0.34	0.31	0.49	0.52	0.3	0.14	0.41	0.37	0.1	-0.031	0.85	1	-0.16	0.27	0.44	0.29
S.F.Ratio	0.096	0.18	0.24	-0.38	-0.29	0.28	0.23	-0.55	-0.36	-0.032	0.14	-0.13	-0.16	1	-0.4	-0.58	-0.31
perc.alumni	-0.09	-0.16	-0.18	0.46	0.42	-0.23	-0.28	0.57	0.27	-0.04	-0.29	0.25	0.27	-0.4	1	0.42	0.49
Expend	0.26	0.12	0.064	0.66	0.53	0.019	-0.084	0.67	0.5	0.11	-0.098	0.43	0.44	-0.58	0.42	1	0.39
Grad.Rate	0.15	0.067	-0.022	0.49	0.48	-0.079	-0.26	0.57	0.42	0.0011	-0.27	0.31	0.29	-0.31	0.49	0.39	1

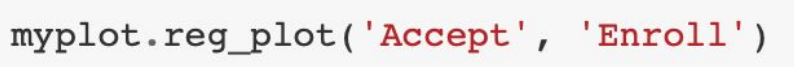
Linear Regression: Analysis Plot Demo



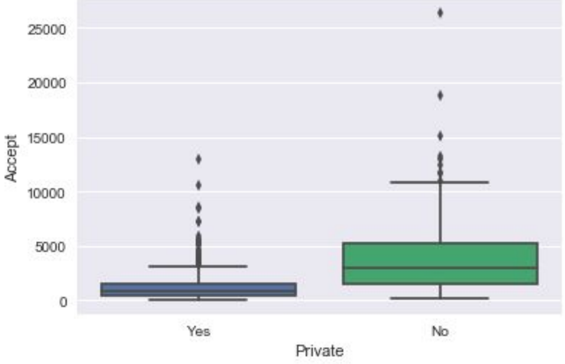
```
myplot.box_plot('Private', 'Accept')
```

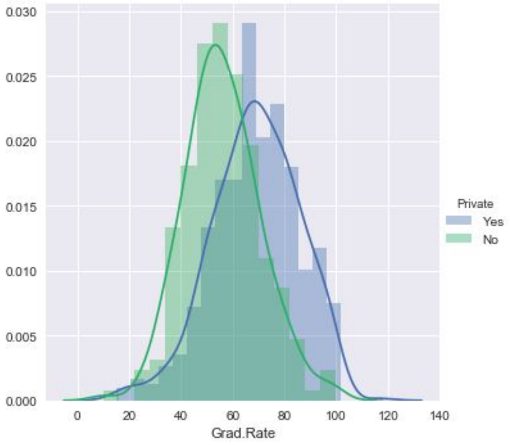


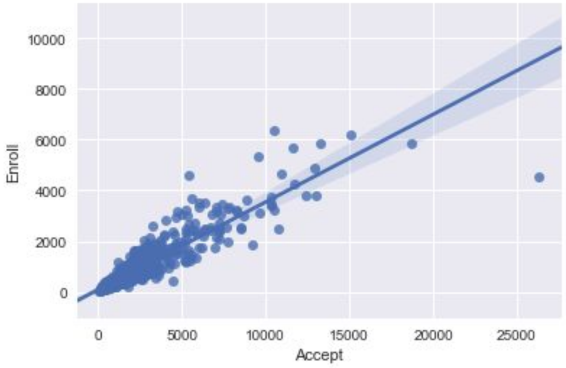
```
myplot.dist_plot('Grad.Rate', 'Private')
```



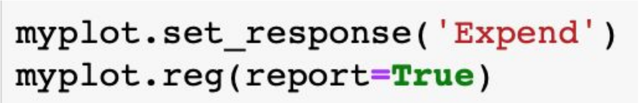
```
myplot.reg_plot('Accept', 'Enroll')
```



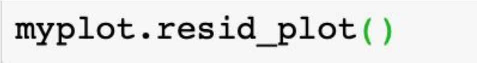




Linear Regression: Analysis Plot Demo



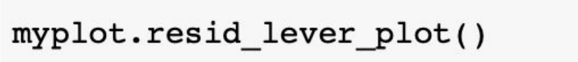
```
myplot.set_response('Expend')
myplot.reg(report=True)
```



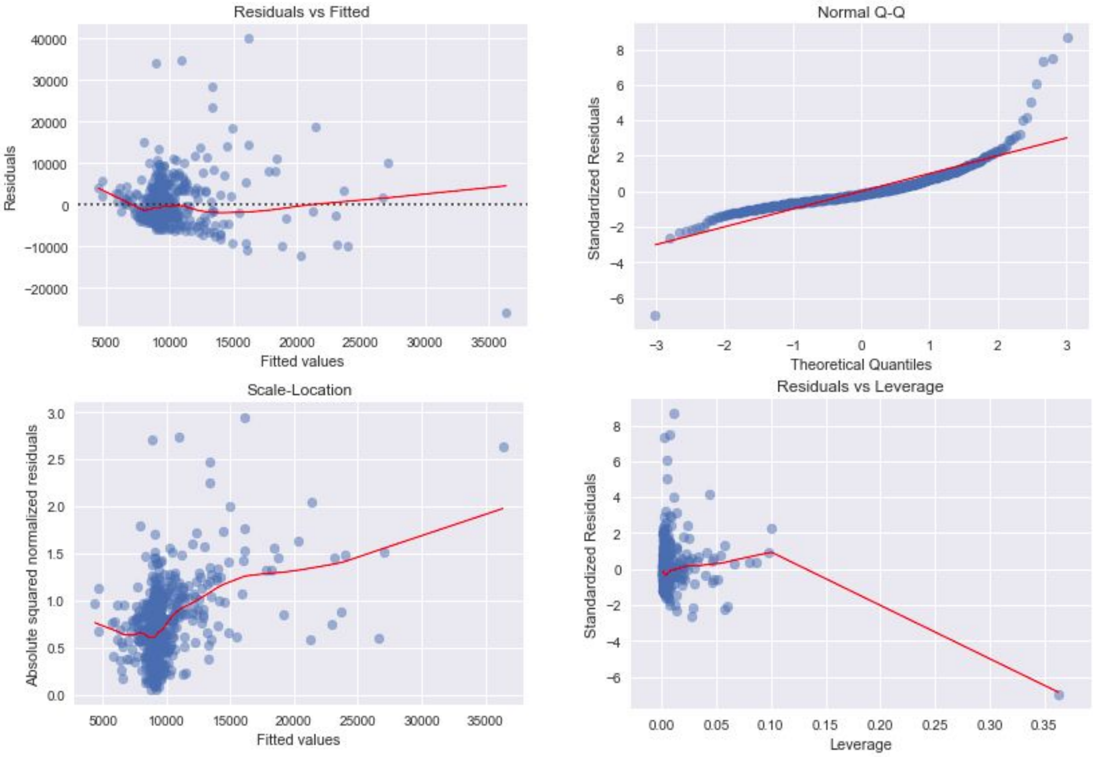
```
myplot.resid_plot()
```

```
myplot.qq_plot()
```

```
myplot.scale_loc_plot()
```



```
myplot.resid_lever_plot()
```



Linear Regression: Dashboard

An interactive dashboard built with Dash

Designed for users to assess linear regression model assumptions

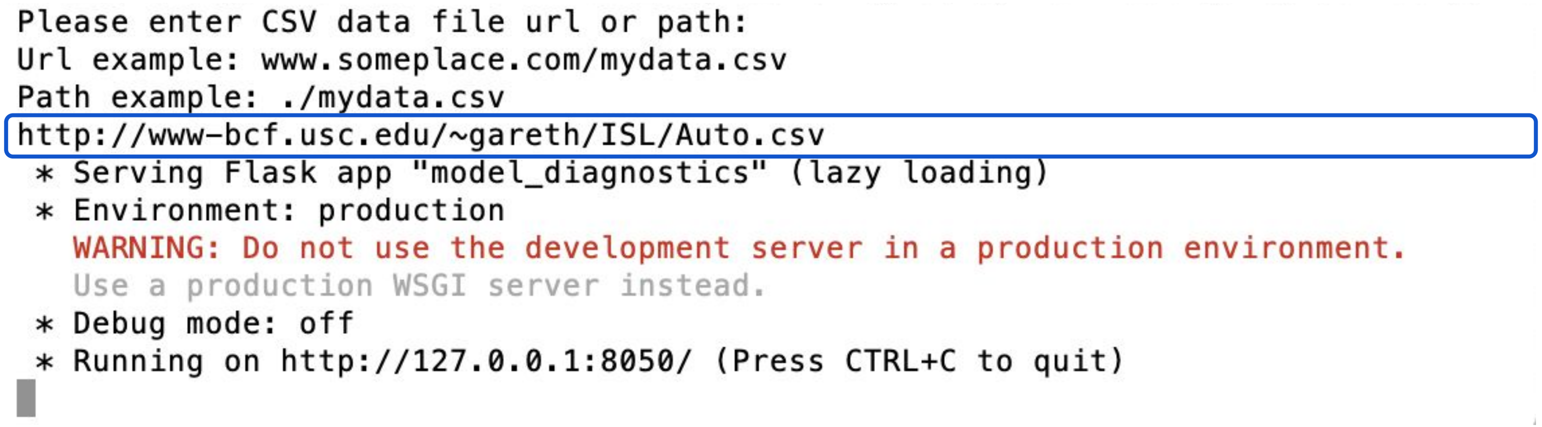
Users need to:

➔ Prepare a CSV data source (online source or local file)
➔ Run Model Diagnostics Tool
➔ Select metrics
➔ Explore plots!

Linear Regression: Dashboard Demo

User Input

- A CSV dataset file address

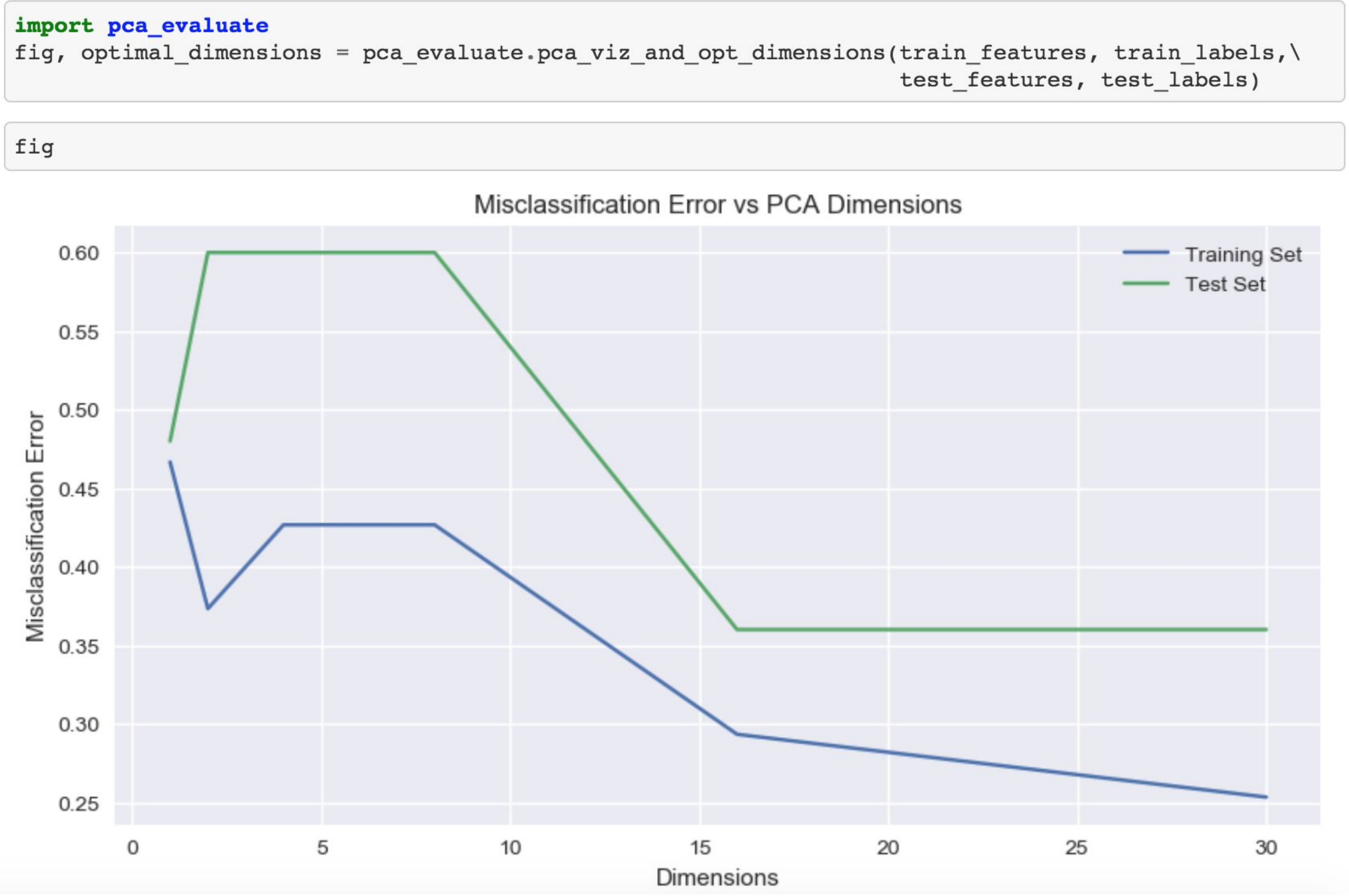


```
Please enter CSV data file url or path:
Url example: www.someplace.com/mydata.csv
Path example: ./mydata.csv
http://www-bcf.usc.edu/~gareth/ISL/Auto.csv
 * Serving Flask app "model_diagnostics" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

PCA Evaluation

- Principal Component Analysis
  - Dimensionality reduction
  - Aiding in data visualization
  - Preprocessing step for subsequent supervised learning
- Given any dataset, one call to the betas pca library will determine the most optimal number of dimensions to use
- Generates a plot to visualize how the misclassification errors change for the test and training sets for the given data source
- Note: for this to work, one would need the response variable as well
- Unsupervised learning optimized using supervised learning metrics

PCA Evaluation Demo I

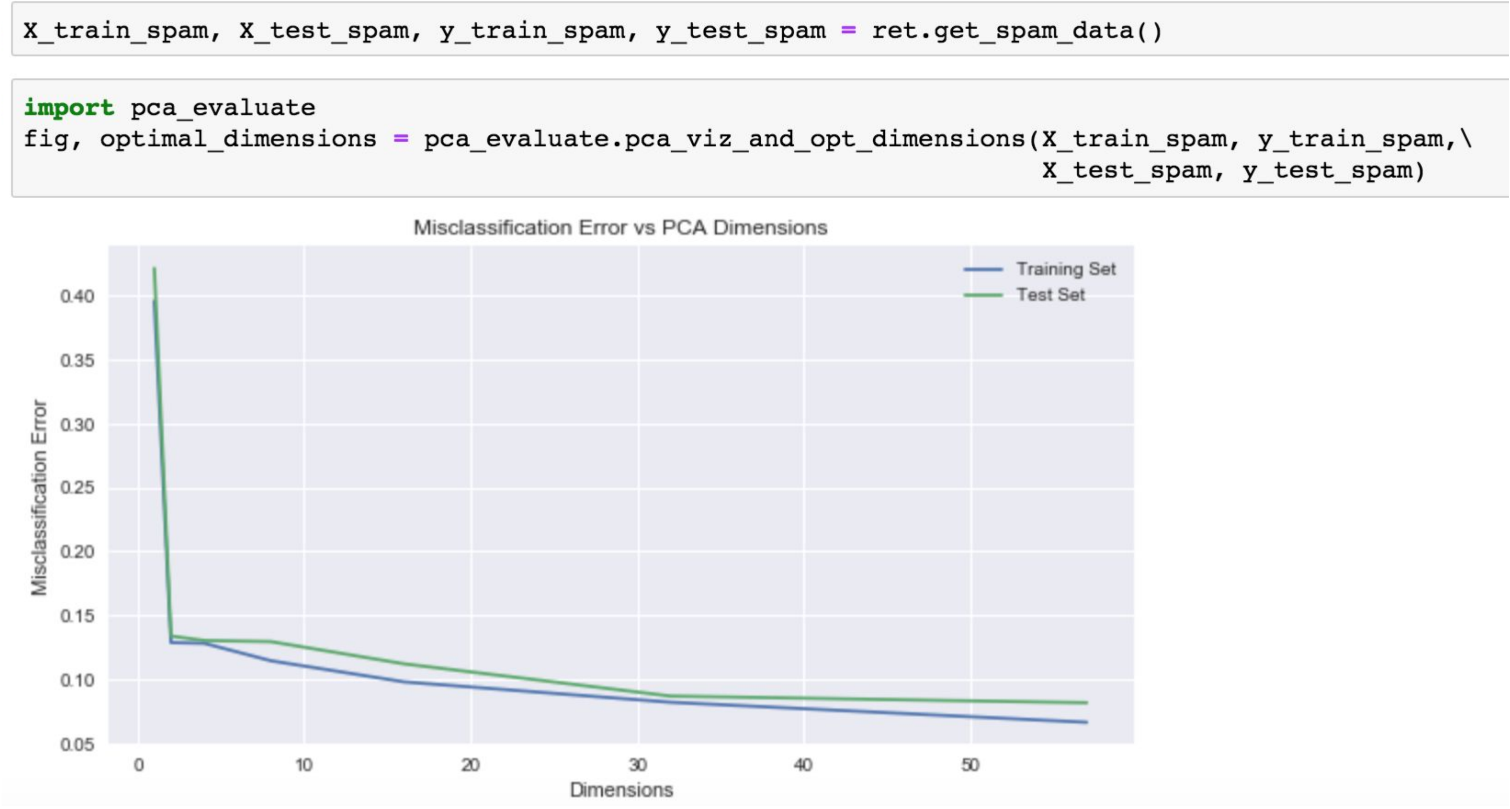


```
import pca_evaluate
fig, optimal_dimensions = pca_evaluate.pca_viz_and_opt_dimensions(train_features, train_labels,\
                                                                  test_features, test_labels)
```

```
fig
```

PCA Evaluation Demo II

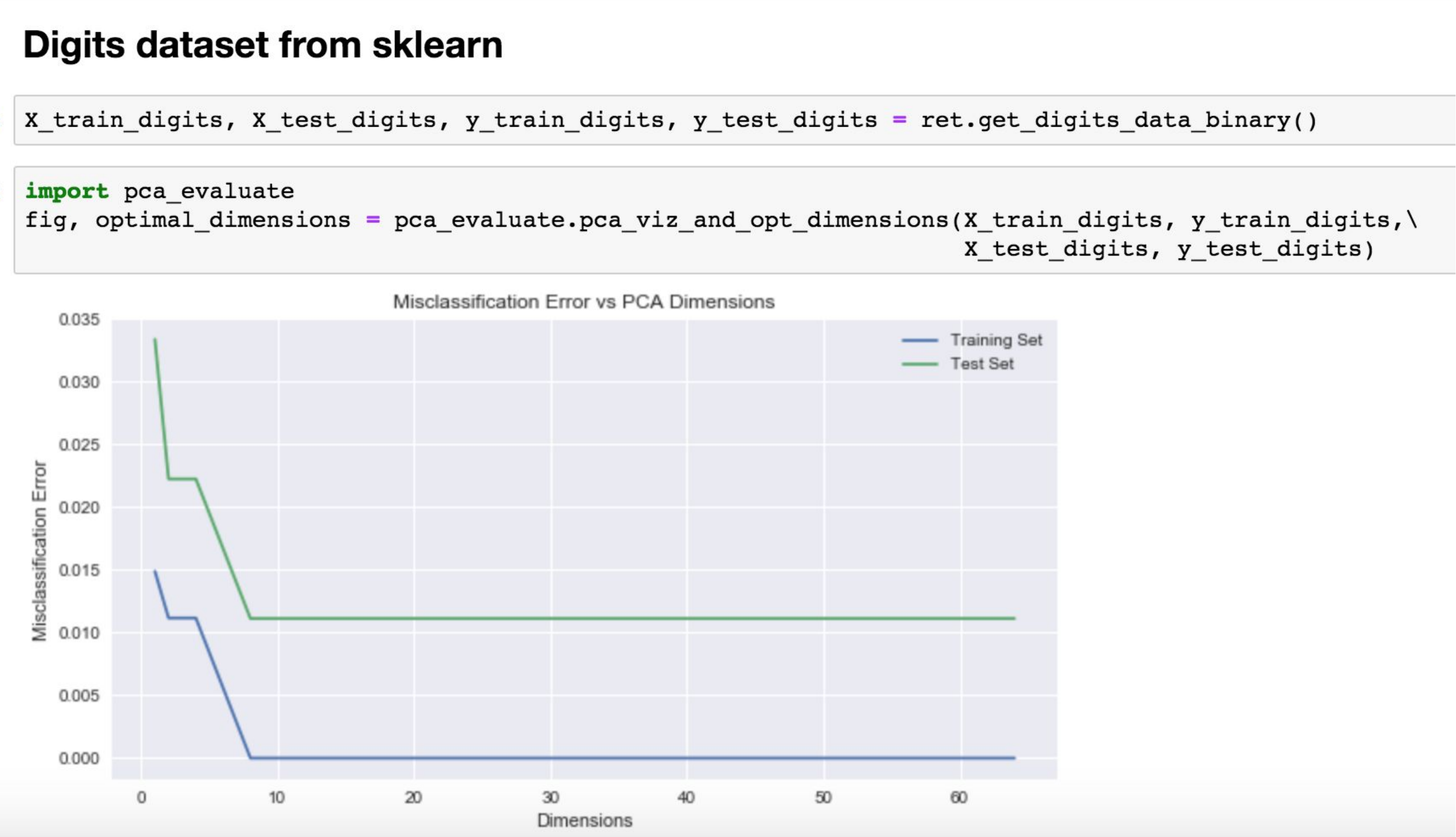
Spam dataset from ESL



```
X_train_spam, X_test_spam, y_train_spam, y_test_spam = ret.get_spam_data()
```

```
import pca_evaluate
fig, optimal_dimensions = pca_evaluate.pca_viz_and_opt_dimensions(X_train_spam, y_train_spam,\
                                                                   X_test_spam, y_test_spam)
```

PCA Evaluation Demo III



Digits dataset from sklearn

```
X_train_digits, X_test_digits, y_train_digits, y_test_digits = ret.get_digits_data_binary()
```

```
import pca_evaluate
fig, optimal_dimensions = pca_evaluate.pca_viz_and_opt_dimensions(X_train_digits, y_train_digits,\
                                                                 X_test_digits, y_test_digits)
```

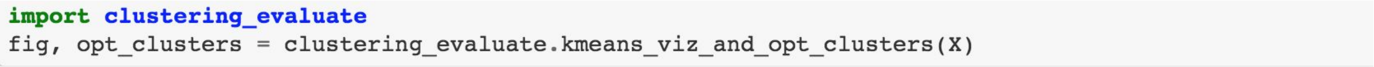
Clustering Evaluation

- Given a dataset of predictors without any response variables, the clustering library determines the most optimal number of clusters to divide the data into

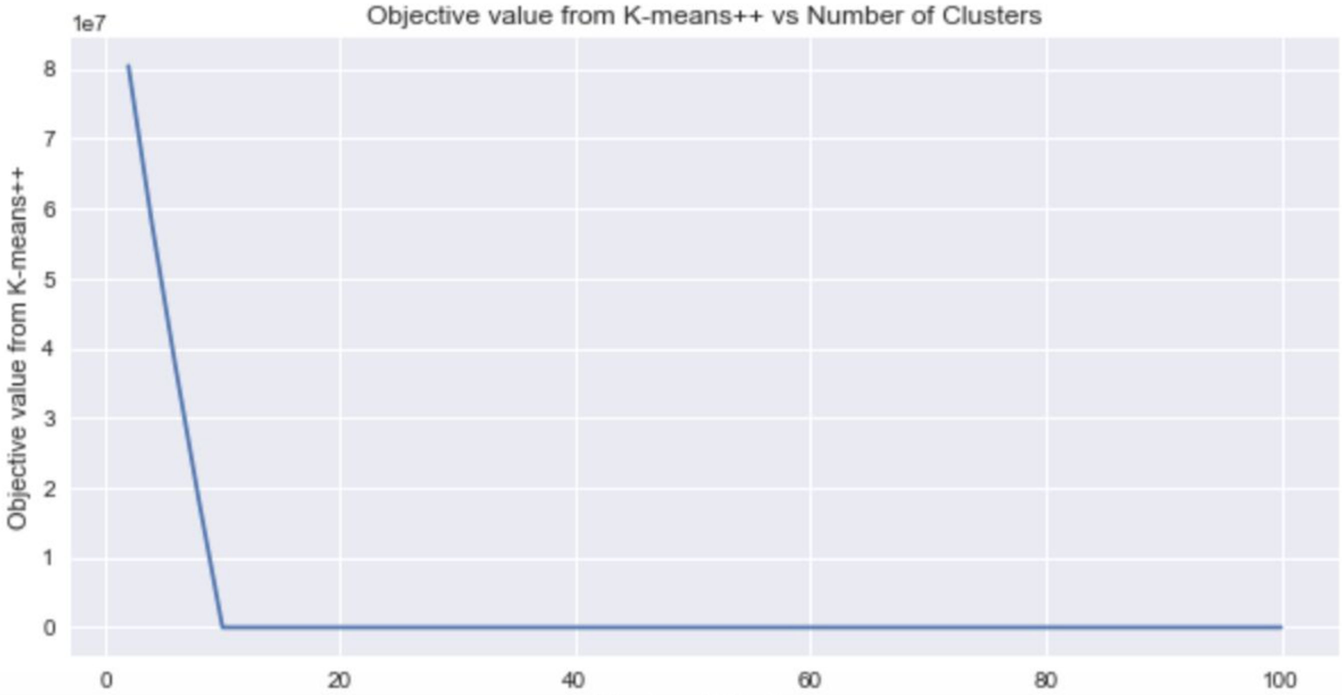
- Unsupervised learning task that uses the inertia/cost for any number of clusters for the given dataset

- Uses k-means++ for a more optimal initialization of the clustering algorithm

Clustering Evaluation Demo



```
import clustering_evaluate
fig, opt_clusters = clustering_evaluate.kmeans_viz_and_opt_clusters(X)
```



Lessons Learned

Lessons Learned

- Version control
  - Add branches
  - Pull requests
- Write unit tests
- Bokeh, Dash
- pip installable package
- Determine the project’s scope as a primary step

Future Work

- Improve the flexibility and extensibility of *betas* library
    - Add more customized arguments to the binary score plots and regression analysis plots, with automatically adjustment in figure scale
    - Addition to Kernel PCA to the PCA module
    - Make the clustering package more customizable, with inclusion of hierarchical clustering, spectral clustering, etc.
- Employ relevant representative datasets, as we want our work to be data agnostic
- Make our package conda installable