

# Final Presentation



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
  - Familiar with: Python, 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 **v1.0** license **MIT** contributors **5** code size **68.4 kB**



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

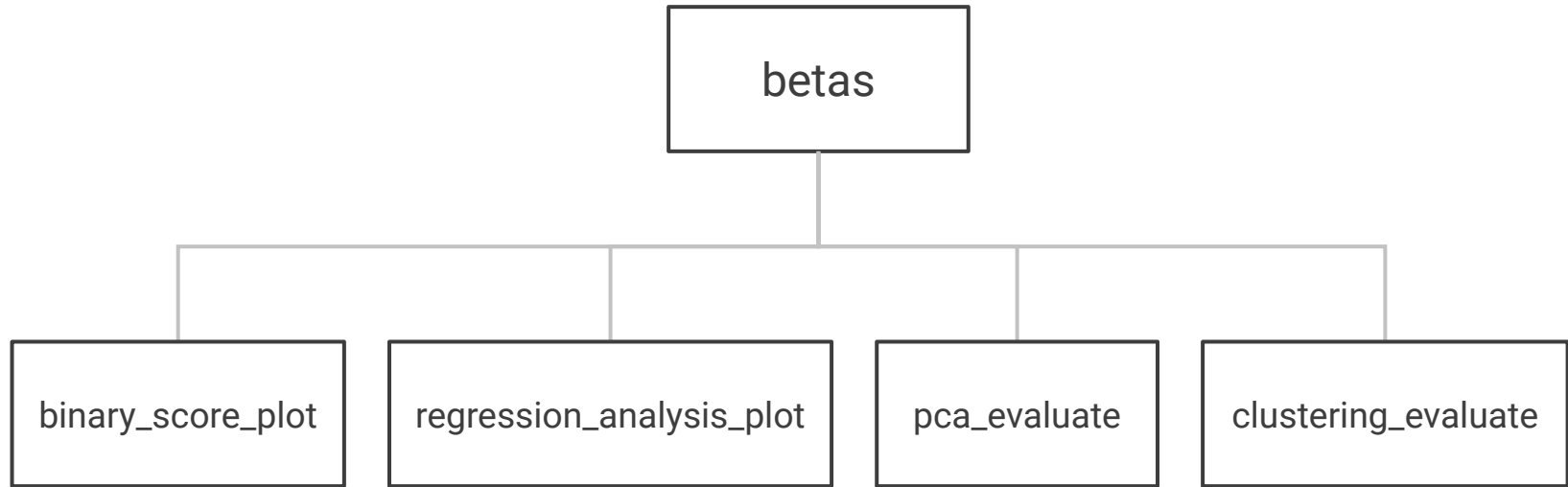
- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Private use

## Limitations

- ✗ Liability
- ✗ Warranty

# betas Library

Individual modules support evaluating four key data science methods



# betas Library

		Unit Test	Documentation	Related demo files
1	binary_score_plot.py	✓	✓	<ul style="list-style-type: none"><li>• test_binary_score_plot.py</li><li>• demo_binary_scoe_plot.ipynb</li><li>• binary_score_diagnostics.py</li></ul>
2	regression_analysis_plot.py	✓	✓	<ul style="list-style-type: none"><li>• test_regression_analysis_plot.py</li><li>• demo_binary_scoe_plot.ipynb</li><li>• binary_score_diagnostics.py</li></ul>
3	pca_evaluate	✓	✓	<ul style="list-style-type: none"><li>• test_pca_evaluate.py</li><li>• demo_pca_evaluate.ipynb</li></ul>
4	clustering_evaluate	✓	✓	<ul style="list-style-type: none"><li>• test_clustering_evaluate.py</li><li>• demo_clustering_evaluate.ipynb</li></ul>

# 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

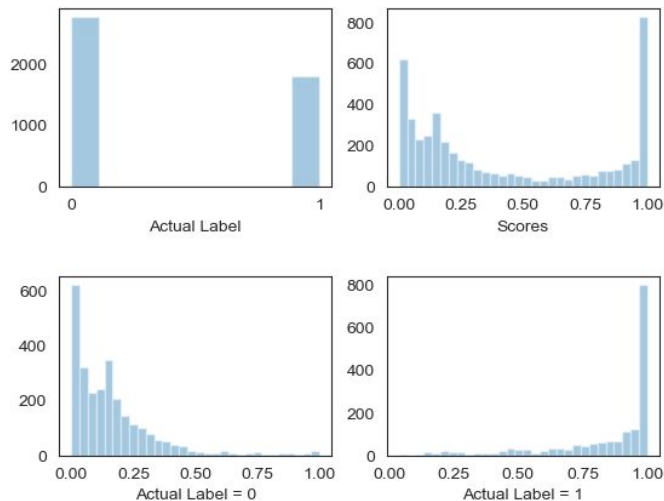
```
from betas import binary_score_plot  
bsp = binary_score_plot.BinaryScorePlot(scores, labels, 0.55)
```



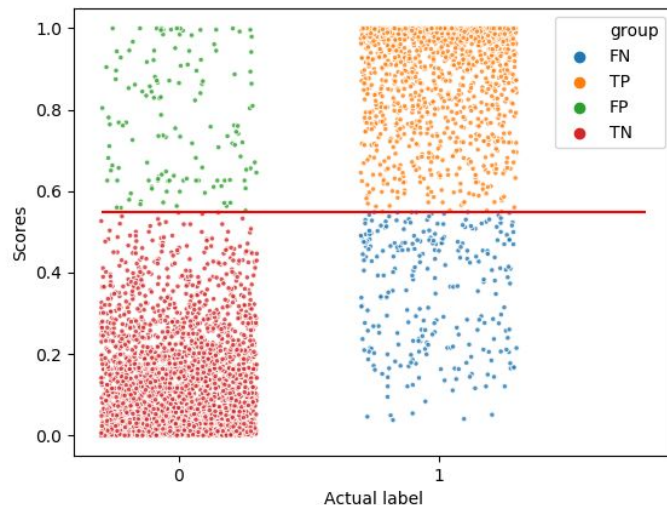
# Binary Classification: Binary Score Plot

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

Histograms of Model Scores by Actual Label

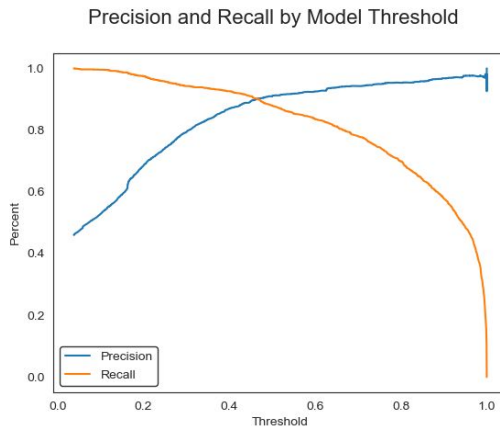
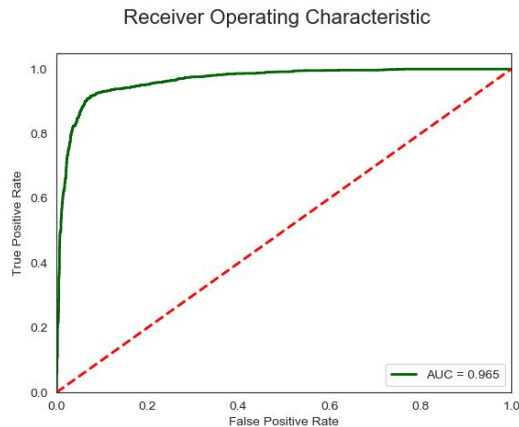


Scatterplot of Model Scores with Threshold = 0.55



# 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*

## Main Methods

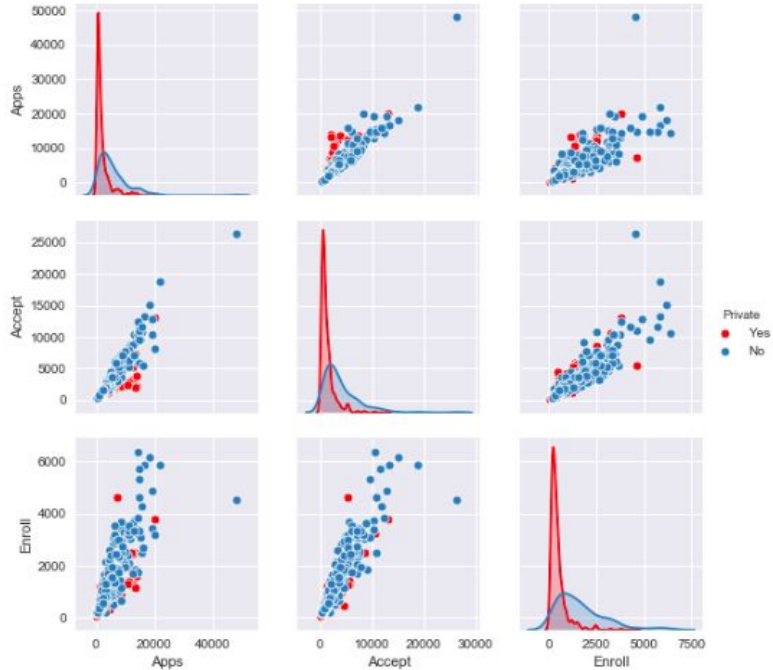
- Scatter matrix plot
- Correlation heatmap
- Box plot
- Distribution plot
- Scatter plot with regression line
- Basic linear regression model
- Model diagnostics

```
import regression_analysis_plot as plt
```

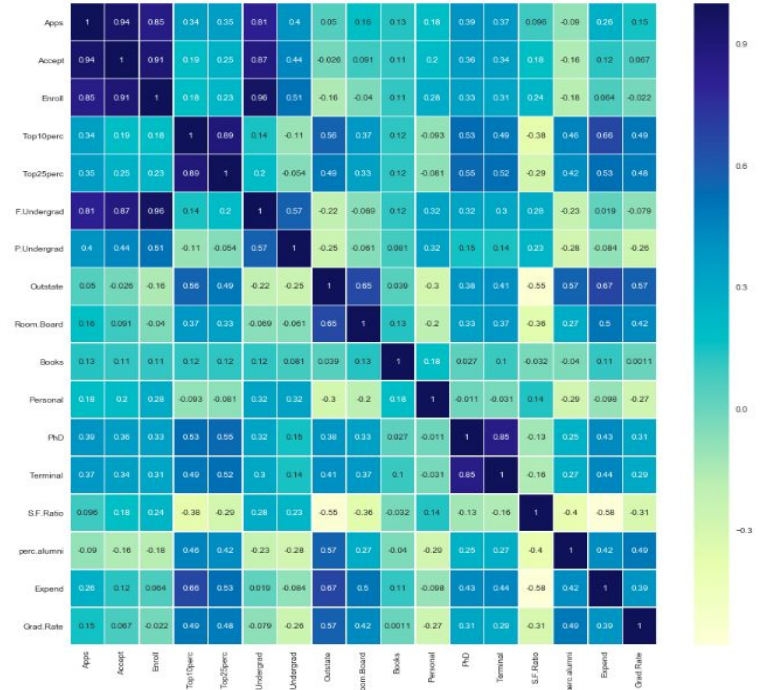
```
myplot = plt.RegressionAnalysisPlot(college)
```

# Linear Regression: Analysis Plot Demo

```
myplot.matrix_plot()
```



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

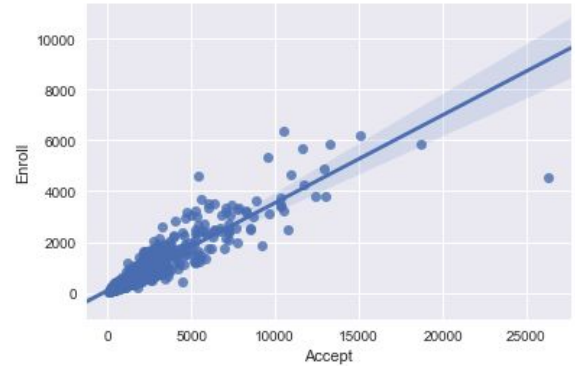
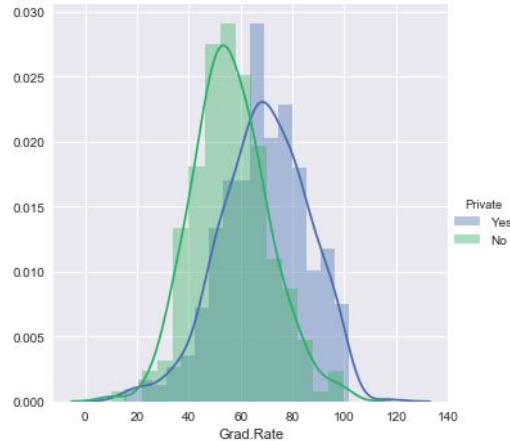
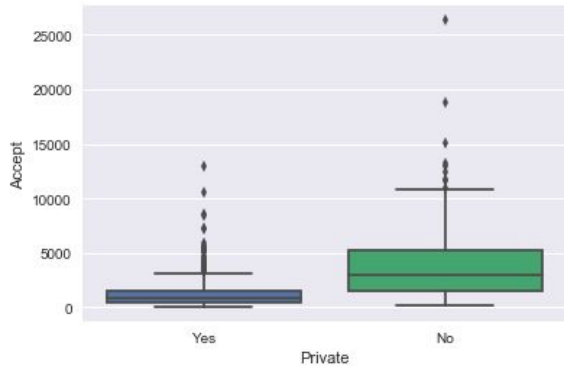


# Linear Regression: Analysis Plot Demo

```
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)
```

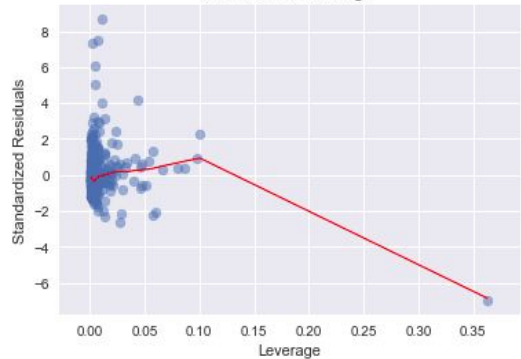
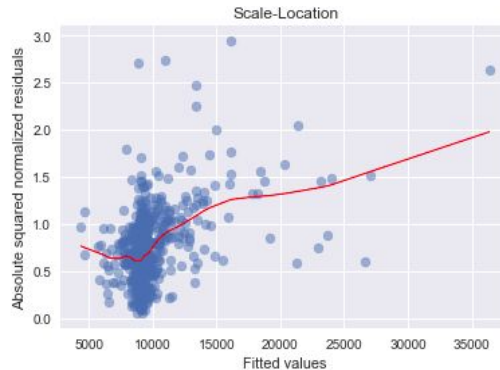
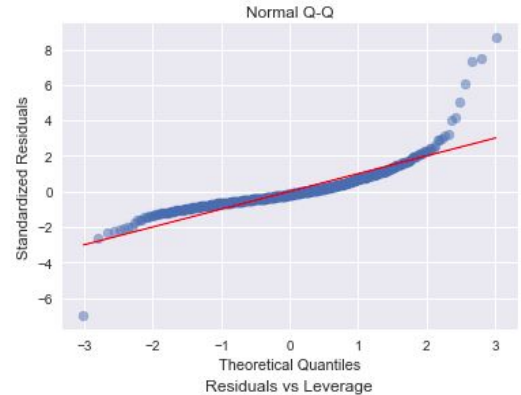
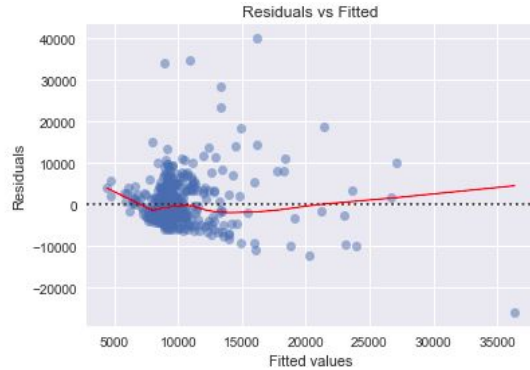


```
myplot.resid_plot()
```

```
myplot.qq_plot()
```

```
myplot.scale_loc_plot()
```

```
myplot.resid_lever_plot()
```



# Linear Regression: Dashboard

An interactive dashboard for users to assess linear regression model

User needs 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

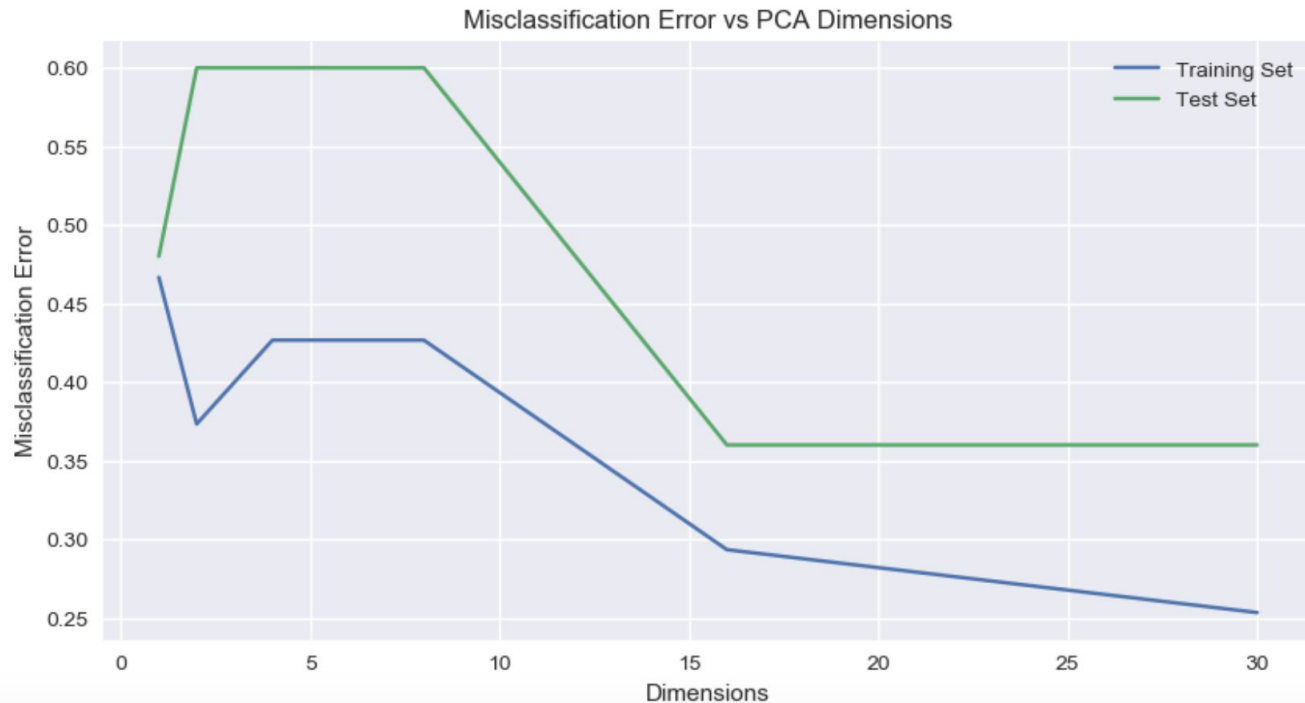
\* Running on `http://127.0.0.1:8050/` (Press CTRL+C to quit)

# PCA Evaluation

- Principal Component Analysis
  - Dimensionality reduction
  - Aiding in data visualization
- 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

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

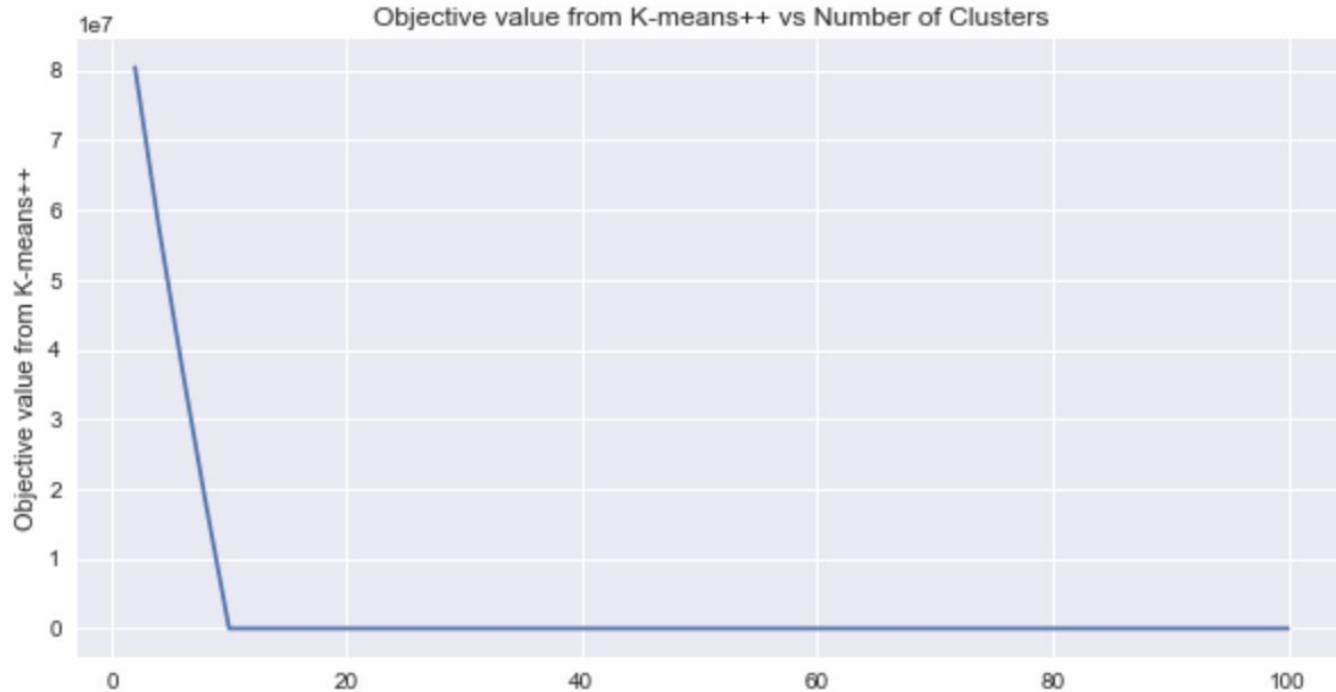


# Clustering Evaluation

- 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 and Future Works

## Lessons Learned

- Version control
  - Add branches
  - Pull requests
- Write unit tests
- Bokeh, Dash
- pip installable package

## Future Works

-