Project Summary

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

Outline

- Project background
- Data Sources
- Project Structure
- Software & License
- Binary Classification
- Linear Regression Analysis
- PCA Evaluation
- Clustering Evaluation

Project Background

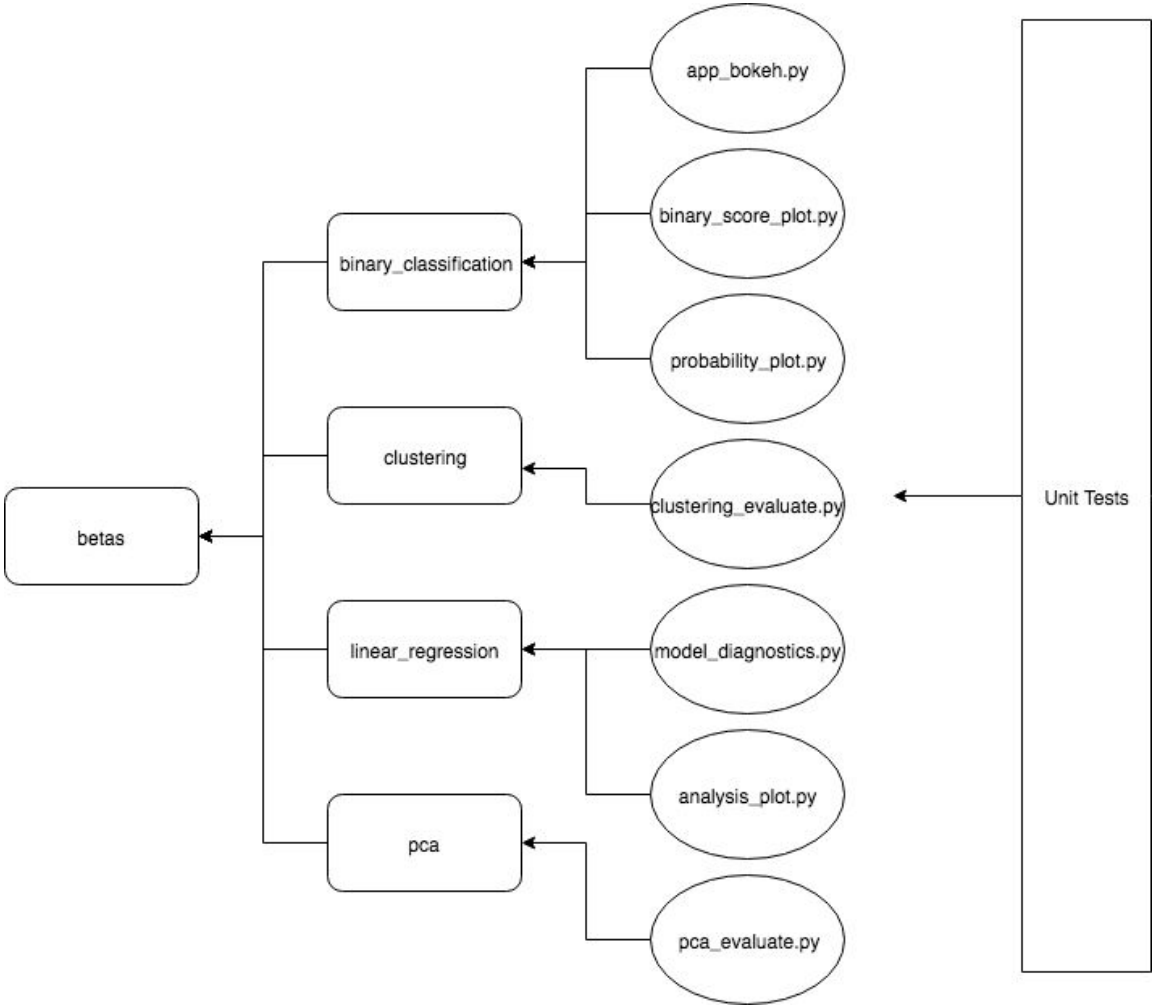
- Betas is a simple and convenient visualization tool for data scientists or analysts who:
    - Do not want to generate custom plots for analyzing model scores or model assumptions
    - Use ML models without a detailed understanding of how the models work and would like to rely on a pre-configured solution for applying commonly used methods
- Betas is pip installable, freely available, and easy to use by following our example ipython notebooks.

Data Source

- The Spam dataset
  - Number of instances: 4601 (1813 Spam = 39.4%)
  - Number of attributes: 58 (57 continuous, 1 class label)
- The College dataset from *An Introduction to Statistical Learning*
  - Number of instance: 777
  - Number of attributes: 19 (17 continuous, 2 nominal)

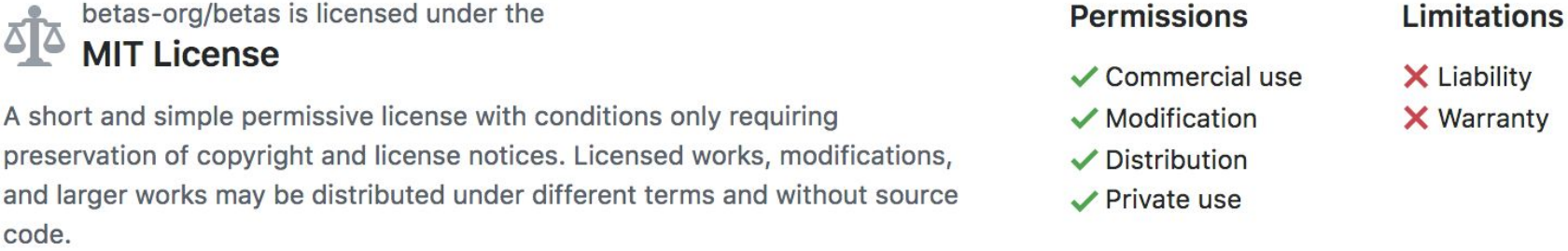
Project Structure

- Betas is an evaluation framework for common machine learning approaches for classification and regression
- Individual modules support evaluating key data science methods
- Unit tests ensure individual modules function properly



Software & License

- Betas is distributed via PyPI and is a pip installable library of methods
- The end user is someone familiar with installing and importing python libraries and following examples of implementing Python functions
- One need not fully understand Statistics or Machine Learning to leverage our methods which make evaluating model scores and assumptions more accessible



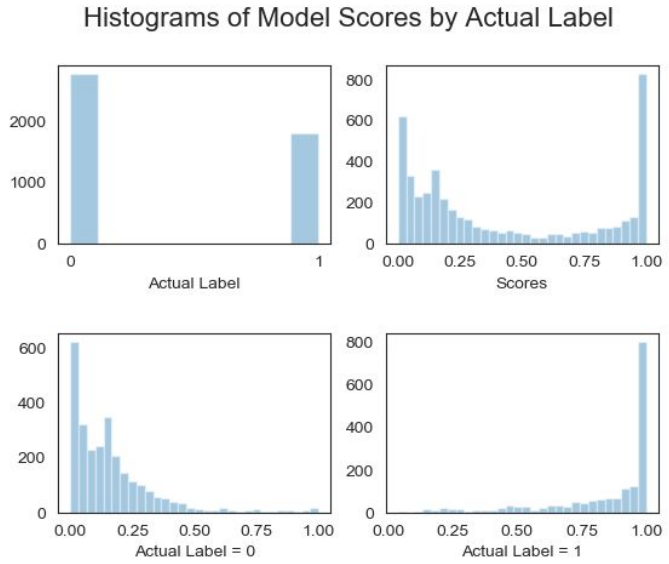
Binary Classification: Binary Score Plot

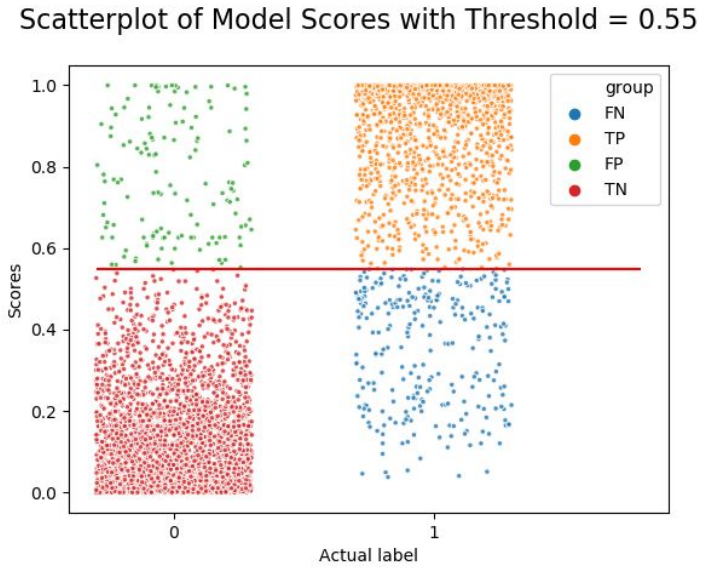
- Histogram of model scores
- Scatterplot of model scores vs. actual labels
- ROC curve
- Precision-recall curve
- Optimal threshold

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

Binary Classification: Binary Score Plot

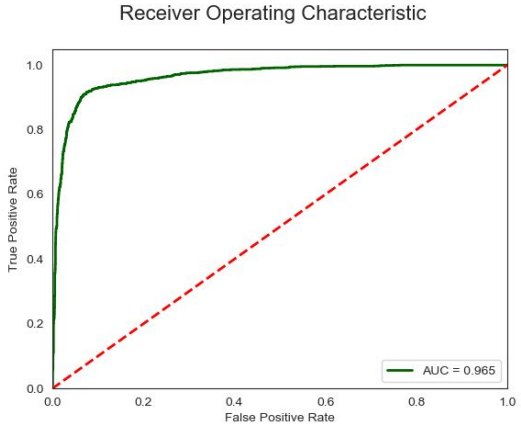
```
bsp.plot_hist()
bsp.plot_jitter()
```

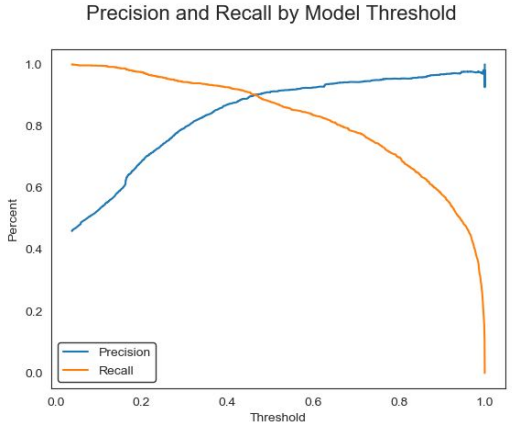




Binary Classification: Binary Score Plot

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






```
bsp.optimal_threshold()
```

```
0.43
```

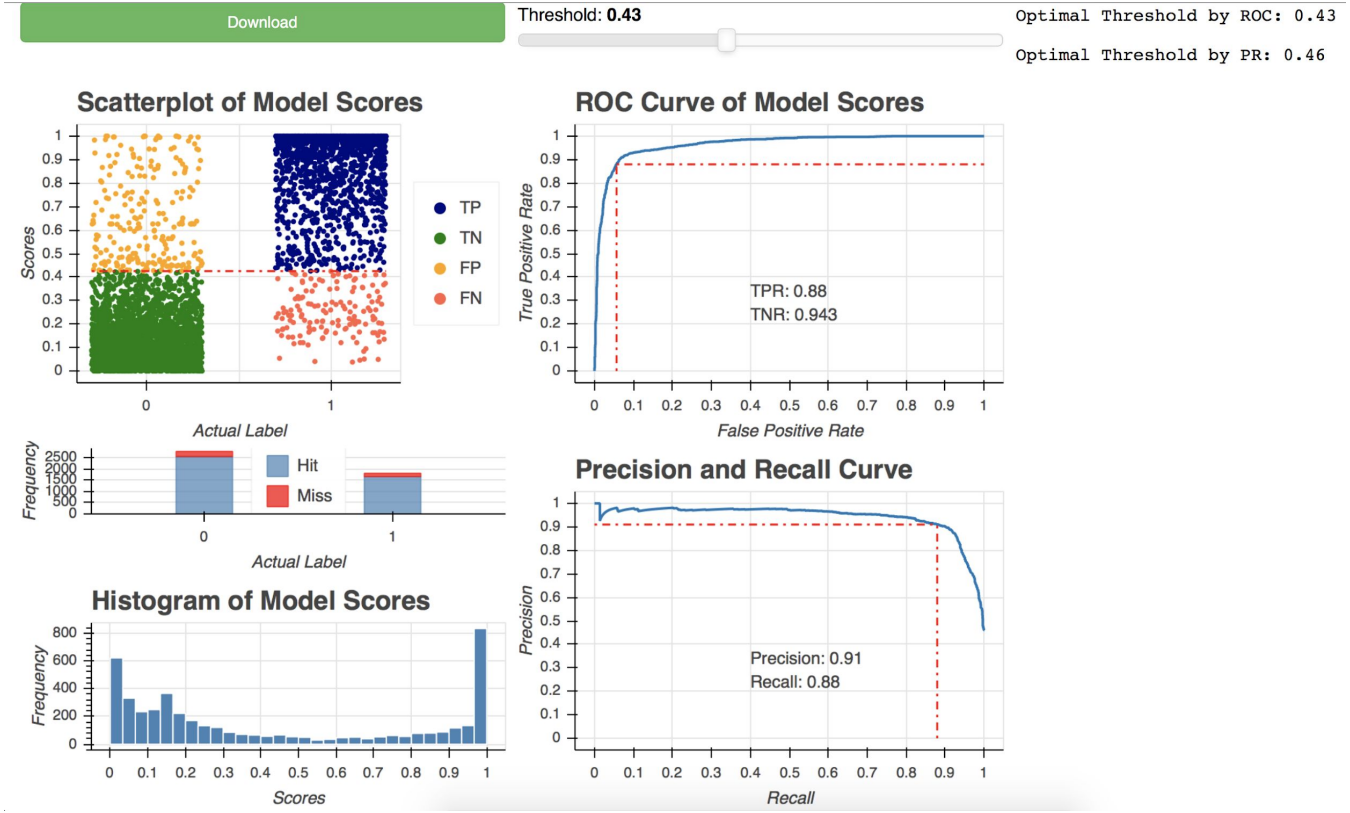
```
bsp.optimal_threshold(by='pr')
```

```
0.46
```

Binary Classification: Visualization Tool

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

Binary Classification: Visualization Tool



Linear Regression: Analysis Plot

User Input

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

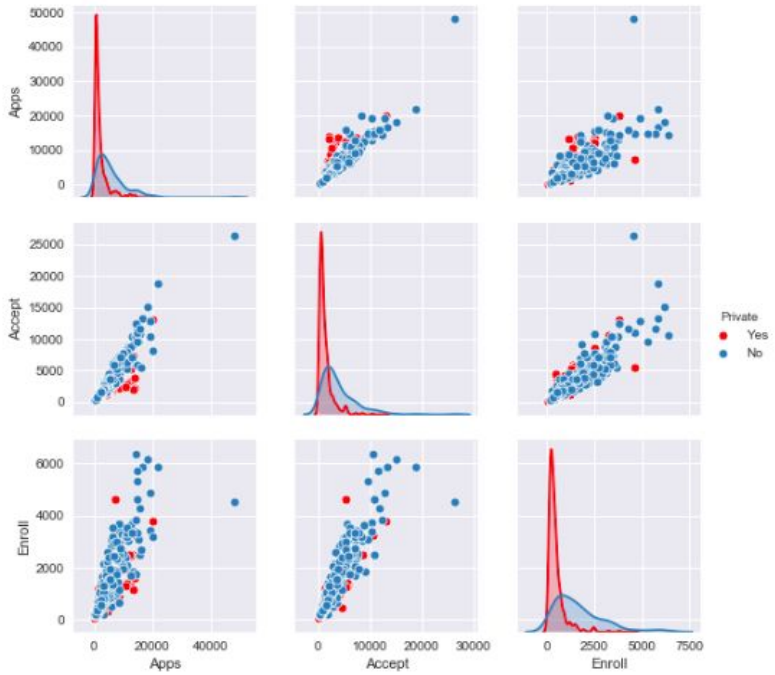
** Predictor(s) and response can be reselect*

Methods

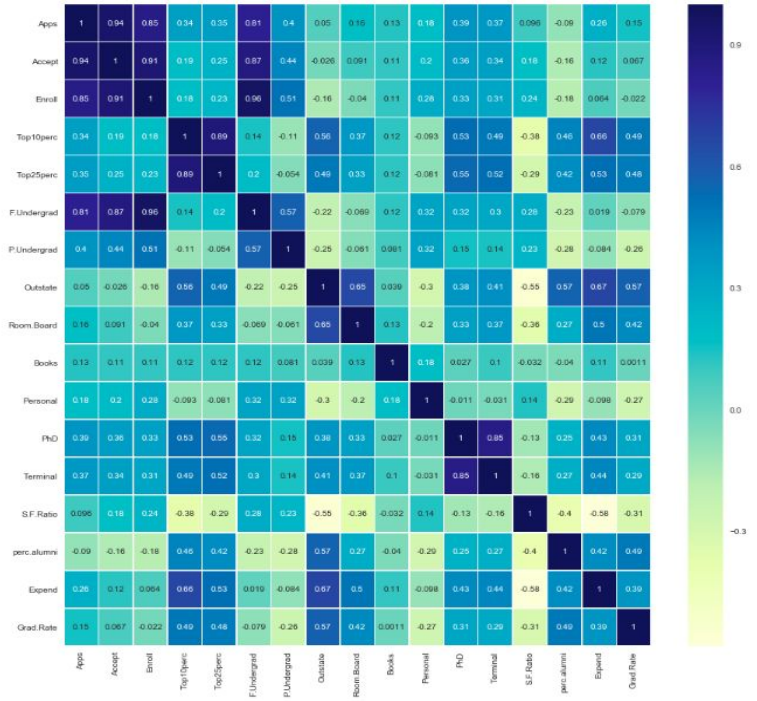
- Get dataframe
- Get/Set Predictors
- Get/Set Response
- Scatter matrix plot
- Correlation heatmap
- Box plot
- Distribution plot
- Scatter plot with regression line
- Basic linear regression model
- Model diagnostics

Linear Regression: Analysis Plot Demo

Scatter matrix plot

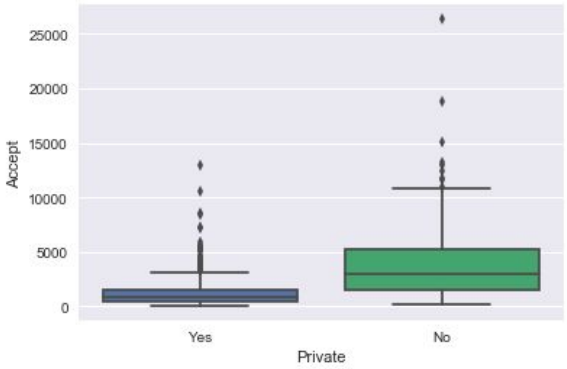


Correlation heatmap

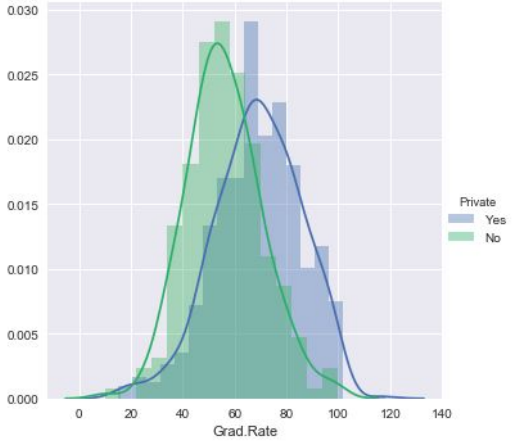


Linear Regression: Analysis Plot Demo

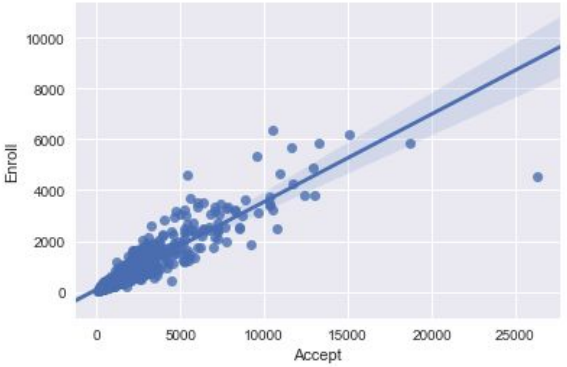
Box plot



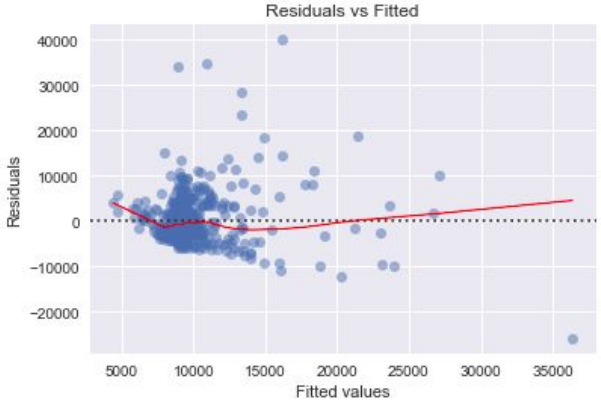
Distribution plot

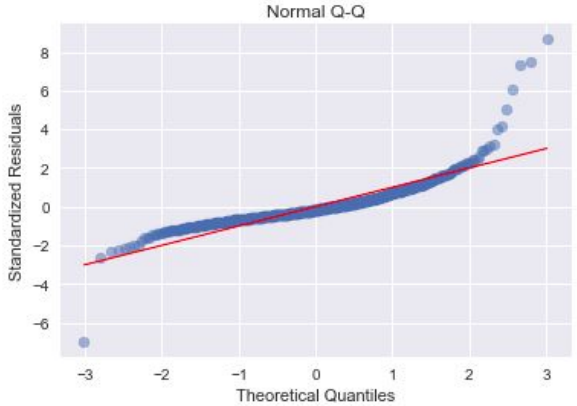


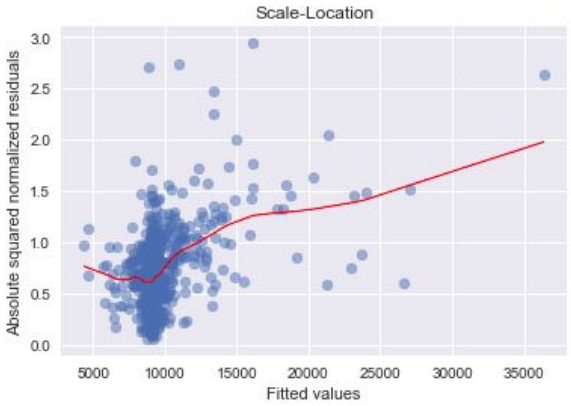
Scatter plot with regression line

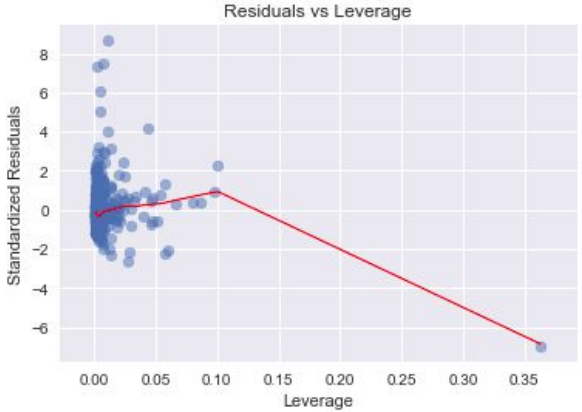


Linear Regression: Analysis Plot Demo









Linear Regression: Model Diagnostics Tool

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!

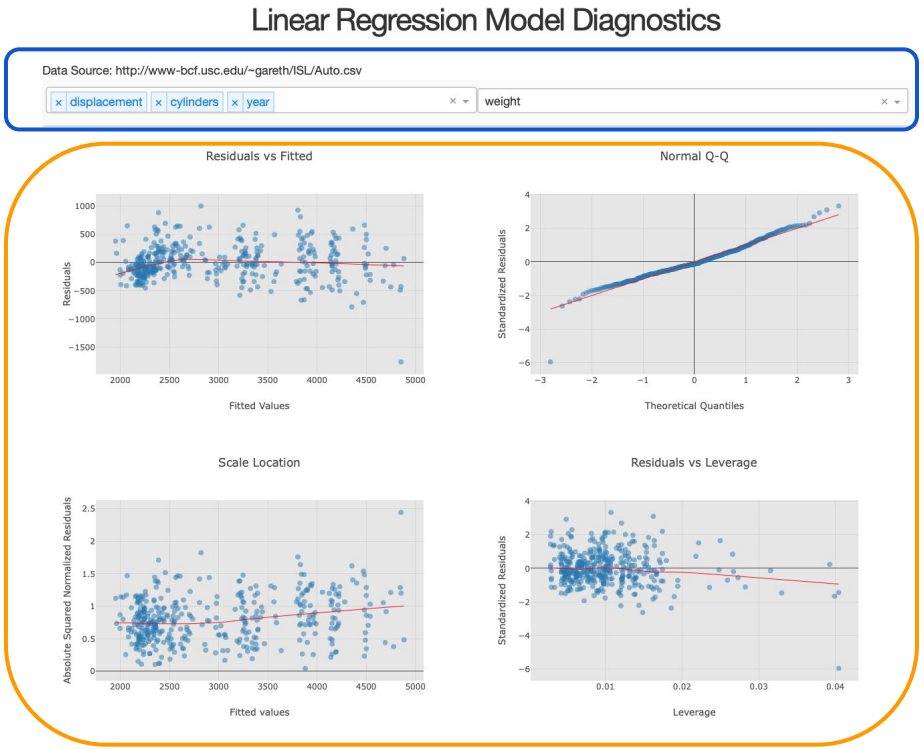
Linear Regression: Model Diagnostics Tool Demo

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)
```

Linear Regression: Model Diagnostics Tool Demo



Make selections

1. Select predictor(s)
2. Select a response

Explore dataset

PCA Evaluation

- Principal Component Analysis: Technique used for dimensionality reduction as well as 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

```
In [8]: import pca_evaluate
        fig, optimal_dimensions = pca_evaluate.pca_viz_and_opt_dimensions(train_features, train_labels,\
                                                                          test_features, test_labels)
```

```
In [9]: fig
```

Out[9]:

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

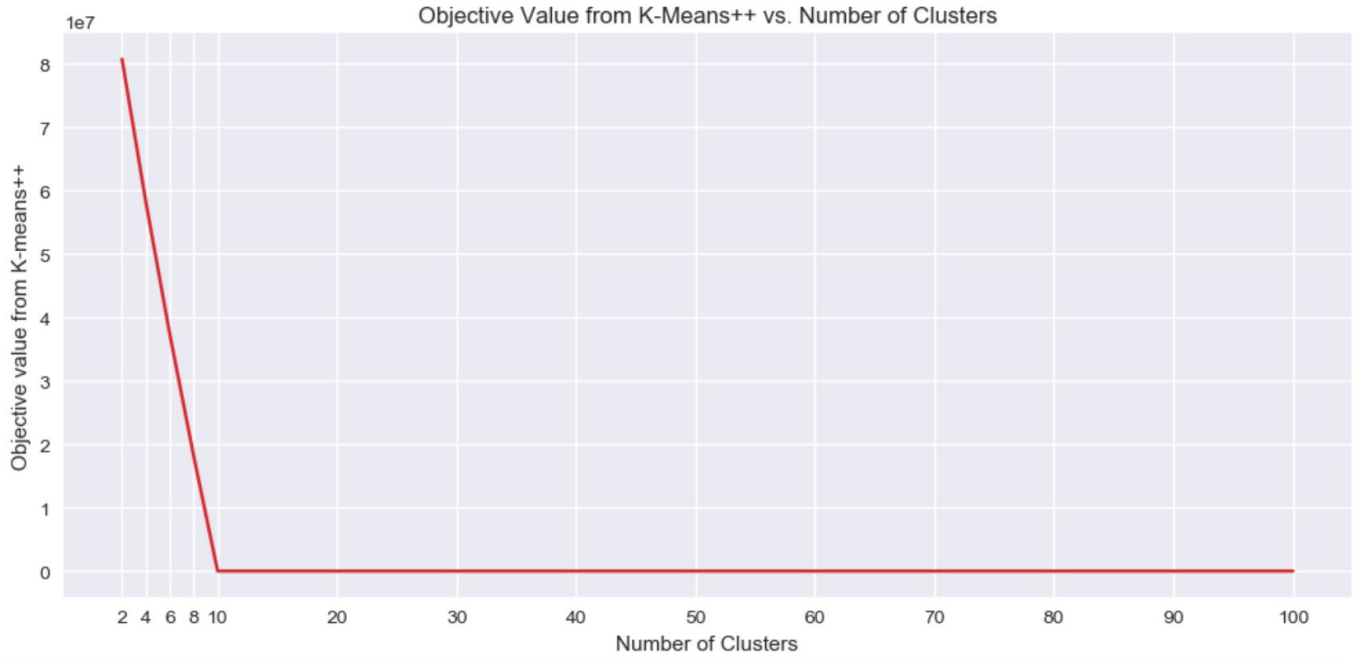
In [2]:
```
import clustering_evaluate
fig, opt_clusters = clustering_evaluate.kmeans_viz_and_opt_clusters(X)
```

In [3]:
```
opt_clusters
```

Out[3]: 10

In [4]:
```
fig
```

Out[4]:



Conclusion

Visualizations for:

- Binary classification
- Linear regression
- PCA
- Clustering

Package with documentation, unit tests, ...

(Depending on how far we get, we'll fill this one up)

Lessons Learned

- Bokeh, Dash
- Pip installable package

Future Work

- Determining the scope of this project, due to open-ended nature
- Employing relevant representative datasets, as we wanted our work to be data agnostic
- Making our package pip installable