GraphLab Cheatsheet

Contents

[GraphLab Cheatsheet 1](#_Toc436928733)

[Classifiers 1](#_Toc436928734)

[Logistic Classifier 1](#_Toc436928735)

[Show ROC Curve for Classifier 2](#_Toc436928736)

[Environment 2](#_Toc436928737)

[Plot into the iPython notebook 2](#_Toc436928738)

[Images 2](#_Toc436928739)

[Show an image 2](#_Toc436928740)

[Plotting 3](#_Toc436928741)

[Box Whisker Plot 3](#_Toc436928742)

[Scatter Plot 3](#_Toc436928743)

[Regression 4](#_Toc436928744)

[Create a regression model 4](#_Toc436928745)

[Coefficients of a regression model 4](#_Toc436928746)

[SFrames 4](#_Toc436928747)

[Add a column to an Sframe 4](#_Toc436928748)

[Apply a function to an Sframe column 4](#_Toc436928749)

[Convert a Dictionary to an SFrame 4](#_Toc436928750)

[Extract a column into an Sarray 5](#_Toc436928751)

[Max of a column 5](#_Toc436928752)

[Mean of a column 5](#_Toc436928753)

[Random partition of an Sframe 5](#_Toc436928754)

[Sort an SFrame 5](#_Toc436928755)

[Summary of a column 5](#_Toc436928756)

[Visualize columns 7](#_Toc436928757)

[Text Analytics 7](#_Toc436928758)

[Word Count 7](#_Toc436928759)

# Classifiers

## Logistic Classifier

sentiment\_model = graphlab.logistic\_classifier.create(train\_data,

target='sentiment',

features=['word\_count'],

validation\_set=test\_data)

## Show ROC Curve for Classifier

sentiment\_model = graphlab.logistic\_classifier.create(train\_data,

target='sentiment',

features=['word\_count'],

validation\_set=test\_data)

sentiment\_model.evaluate(test\_data, metric='roc\_curve')

sentiment\_model.show(view='Evaluation')

# Environment

## Plot into the iPython notebook

graphlab.canvas.set\_target('ipynb')

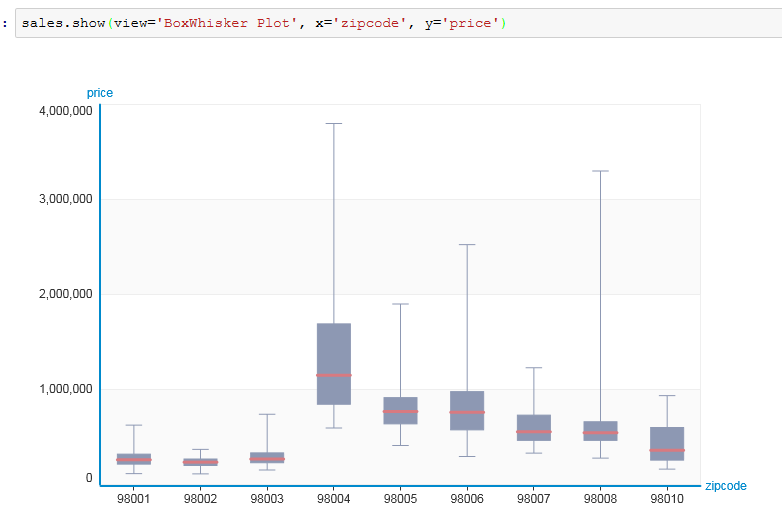
# Images

## Show an image

<img src="house-1925069082.jpg">

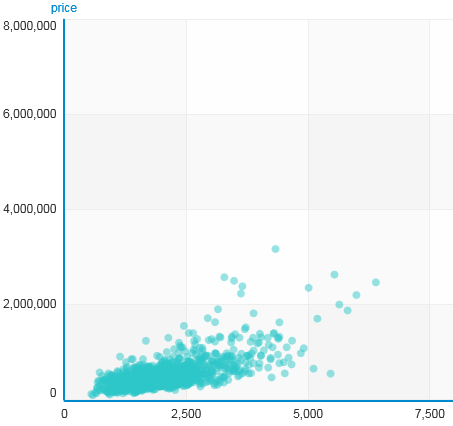
# Plotting

## Box Whisker Plot



## Scatter Plot

sales.show(view="Scatter Plot", x="sqft\_living", y="price")



# Regression

## Create a regression model

sqft\_model = graphlab.linear\_regression.create(train\_data, target='price', features=['sqft\_living'])

## Coefficients of a regression model

sqft\_model.get('coefficients')

# Sarrays

## Average of an SArray

avg\_price\_2 = prices.mean()

## Length of an SArray

num\_houses = prices.size()

## Multiply Two Sarrays Element-Wise

prices\_squared = prices\*prices

## Sum of an SArray

sum\_prices = prices.sum()

# SFrames

## Add a column to an Sframe

sf['Full Name'] = sf['First Name'] + ' ' + sf['Last Name']

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First Name** | **Last Name** | **Country** | **age** | **Full Name** |
| Bob | Smith | United States | 24 | Bob Smith |
| Alice | Williams | Canada | 23 | Alice Williams |
| Malcolm | Jone | England | 22 | Malcolm Jone |

## Apply a function to an Sframe column

def transform\_country(country):

if country == 'USA':

return 'United States'

else:

return country

sf['Country'] = sf['Country'].apply(transform\_country)

## Convert a Dictionary to an SFrame

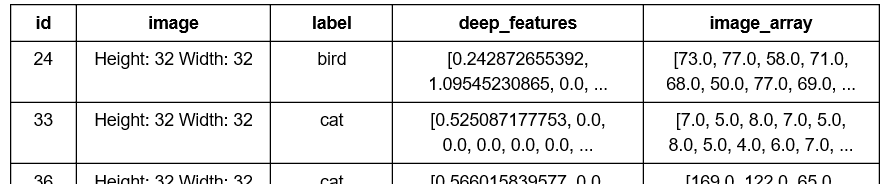
obama\_word\_count\_table = obama[['word\_count']].stack('word\_count', new\_column\_name = ['word','count'])

Note: obama[[‘word\_count’]] is a dict

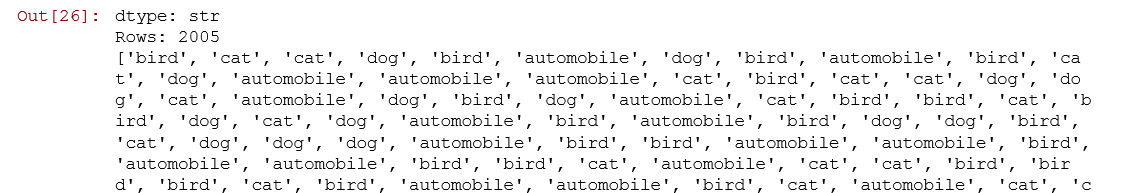
Returns a graphlab.data\_structures.sframe.Sframe

## Extract a column into an Sarray

image\_train.head()



image\_train['label']



## Max of a column

sf['age'].max()

## Mean of a column

sf['age'].mean()

## Random partition of an Sframe

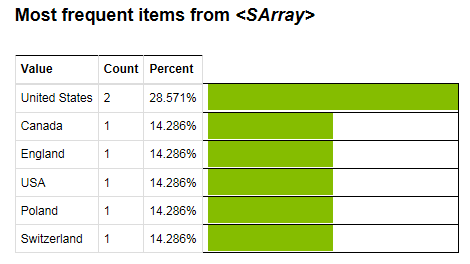
train\_data,test\_data = sales.random\_split(.8,seed=0)

## Sort an SFrame

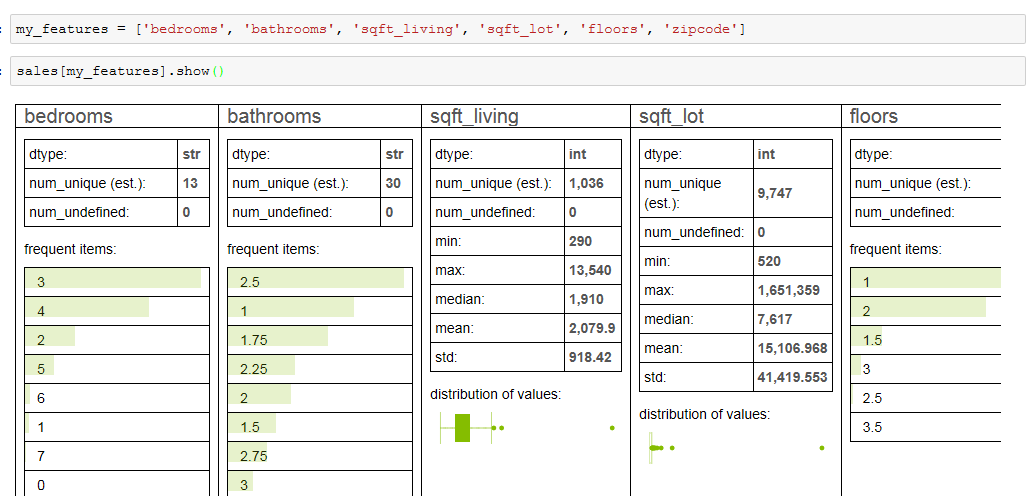
giraffe\_reviews = giraffe\_reviews.sort('predicted\_sentiment', ascending=False)

## Summary of a column

sf['Country'].show()



## Visualize columns



# Text Analytics

## Word Count

products['word\_count'] = graphlab.text\_analytics.count\_words(products['review'])

