

Machine learning for facial recognition

📅 January 21, 2015 (<http://efavdb.com/machine-learning-for-facial-recognition-3/>) 👤 Jonathan Landy (<http://efavdb.com/author/jslandy/>) 📁 Case studies (<http://efavdb.com/category/case-studies/>), Guest posts (<http://efavdb.com/category/guest-posts/>)

A guest post, contributed by Damien Ramunno-Johnson ([LinkedIn](https://www.linkedin.com/profile/view?id=60223336&authType=NAME_SEARCH&authToken=LOV_&locale=en_US&trk=tyah2&trkInfo=tarId%3A1420748440448%2Ctas%3Adamien%2Cidx%3A1-1-1) (https://www.linkedin.com/profile/view?id=60223336&authType=NAME_SEARCH&authToken=LOV_&locale=en_US&trk=tyah2&trkInfo=tarId%3A1420748440448%2Ctas%3Adamien%2Cidx%3A1-1-1), [bio-sketch](http://www.efavdb.com/about) (<http://www.efavdb.com/about>))

Follow @efavdb

Follow us on twitter for new submission alerts!

Introduction

The ability to identify faces is a skill that people develop very early in life and can apply almost effortlessly. One reason for this is that our brains are very well adapted for pattern recognition. In contrast, facial recognition can be a somewhat difficult problem for computers.

Today, given a full frontal image of a face, computer facial recognition software works well. However, problems can arise given large camera angles, poor lighting, or exaggerated facial expressions: Computers have a ways to go before they catch up with us in this arena.

Although facial recognition algorithms remain imperfect, the methods that exist now are already quite useful and are being applied by many different companies. Two examples, first up Facebook: When you upload pictures to their website, it will now automatically suggest names for the people in your photos. This application is well-suited for machine learning for two reasons. First, every tagged photo already uploaded to the site provides labeled examples on which to train an algorithm, and second, people often post full face images in decent lighting. A second example is provided by Google's Android phone OS, which has a face unlock mode. To get this to work, you first have to train your phone by taking images of your face in different lighting conditions and from different angles. After training, the phone can attempt to recognize you. This is another cool application that also often works well.

In this post, we're going to develop our own basic facial learning algorithm. We'll find that it is actually pretty straightforward to set one up that is reasonably accurate. Our post follows and expands upon the tutorial found [here](http://scikit-learn.org/stable/auto_examples/applications/face_recognition.html) (http://scikit-learn.org/stable/auto_examples/applications/face_recognition.html).

Loading packages and data

```
1  from __future__ import print_function
2
3  from time import time
4  import matplotlib.pyplot as plt
5
6  from sklearn.cross_validation import train_test_split
7  from sklearn.datasets import fetch_lfw_people
8  from sklearn.grid_search import GridSearchCV
9  from sklearn.metrics import classification_report
10 from sklearn.metrics import confusion_matrix
11 from sklearn.decomposition import RandomizedPCA
12 from sklearn.svm import SVC
13 import pandas as pd
14 %matplotlib inline
```

The sklearn function [fetch_lfw_people](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html) (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html), imported above, will download the data that we need, if not already present in the faces folder. The dataset we are downloading consists of a set of preprocessed images from [Labeled Faces in the Wild \(LFW\)](http://vis-www.cs.umass.edu/lfw/) (<http://vis-www.cs.umass.edu/lfw/>), a database designed for studying unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web, each labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set.

In our analysis here, we will impose two conditions.

1. First, we will only consider folks that have a minimum of 70 pictures in the data set.
2. We will resize the images so that they each have a 0.4 aspect ratio.

```

1 print('Loading Data')
2 people = fetch_lfw_people(
3     './faces', min_faces_per_person=70, resize=0.4)
4 print('Done!')
5 >>
6 Loading Data
7 Done!

```

The object **people** contains the following data.

1. people.data: a numpy array with the shape(n_samples, h*w), each row corresponds to a unravelled face.
2. people.images: a numpy array with the shape(n_samples, h, w), where each row corresponds to a face. The remaining indices here contain gray-scale values for the pixels of each image.
3. people.target: a numpy array with the shape(n_samples), where each row is the label for the face.
4. people.target_name: a numpy array with the shape(n_labels), where each row is the name for the label.

For the algorithm we will be using, we don't need the relative position data, so we will use the unraveled people.data.

```

1 #Find out how many faces we have, and
2 #the size of each picture from.
3 n_samples, h, w = people.images.shape
4
5 X = people.data
6 n_features = X.shape[1]
7
8 y = people.target
9 target_names = people.target_names
10 n_classes = target_names.shape[0]
11
12 print("Total dataset size:")
13 print("n_images: %d" % n_samples)
14 print("n_features: %d" % n_features)
15 print("n_classes: %d" % n_classes)
16 >>
17 Total dataset size:
18 n_images: 1288
19 n_features: 1850
20 n_classes: 7

```

Looking above we see that our dataset currently has 1288 images. Each image has 1850 pixels, or features. We also have 7 classes, meaning images of 7 different people.

Data segmentation and dimensional reduction

At this point we need to segment our data. We are going to use [train_test_split](http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html) (http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html), which will take care of splitting our data into random training and testing data sets. Next, we note that we have a lot of features and that there are advantages to having fewer: First, the computational cost is reduced. Second, having fewer features reduces the data's dimension which can also reduce the complexity of the model and help avoid overfitting. Instead of dropping individual pixels outright, we will carry out a dimensional reduction via a Principle Component Analysis [PCA](http://en.wikipedia.org/wiki/Principal_component_analysis) (http://en.wikipedia.org/wiki/Principal_component_analysis). PCA works by attempting to represent the variance in the training data with as few dimensions as possible. So instead of dropping features, as we did in our [wearable sensor example](http://efavdb.com/machine-learning-with-wearable-sensors/) (<http://efavdb.com/machine-learning-with-wearable-sensors/>) analysis, here we will compress features together, and then use only the most important feature combinations. When this is done to images, the features returned by PCA are commonly called eigenfaces (some examples are given below).

The function we are going to use to carry out our PCA is [RandomizedPCA](http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.RandomizedPCA.html) (<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.RandomizedPCA.html>). We'll keep the top 150 eigenfaces, and we'll also whiten the data — ie normalize our new, principal component feature set. The goal of whitening is to make the input less redundant. Whitening is performed by rotating into the coordinate space of the principal components, dividing each dimension by square root of variance in that direction (giving the feature unit variance), and then rotating back to pixel space.

```

1  # split into a training and testing set
2  X_train, X_test, y_train, y_test = train_test_split(
3      X, y, test_size=0.25)
4
5  # Compute the PCA (eigenfaces) on the face dataset
6  n_components = 150
7
8  pca = RandomizedPCA(
9      n_components=n_components, whiten=True).fit(X_train)
10
11 eigenfaces = pca.components_.reshape((n_components, h, w))
12 X_train_pca = pca.transform(X_train)

```

Visualizing the eigenfaces

Let's now take a moment to examine the dataset's principal eigenfaces: the set of images that we will project each example onto to obtain independent features. To do this we will use the following helper function to make life easier — visual follows.

```

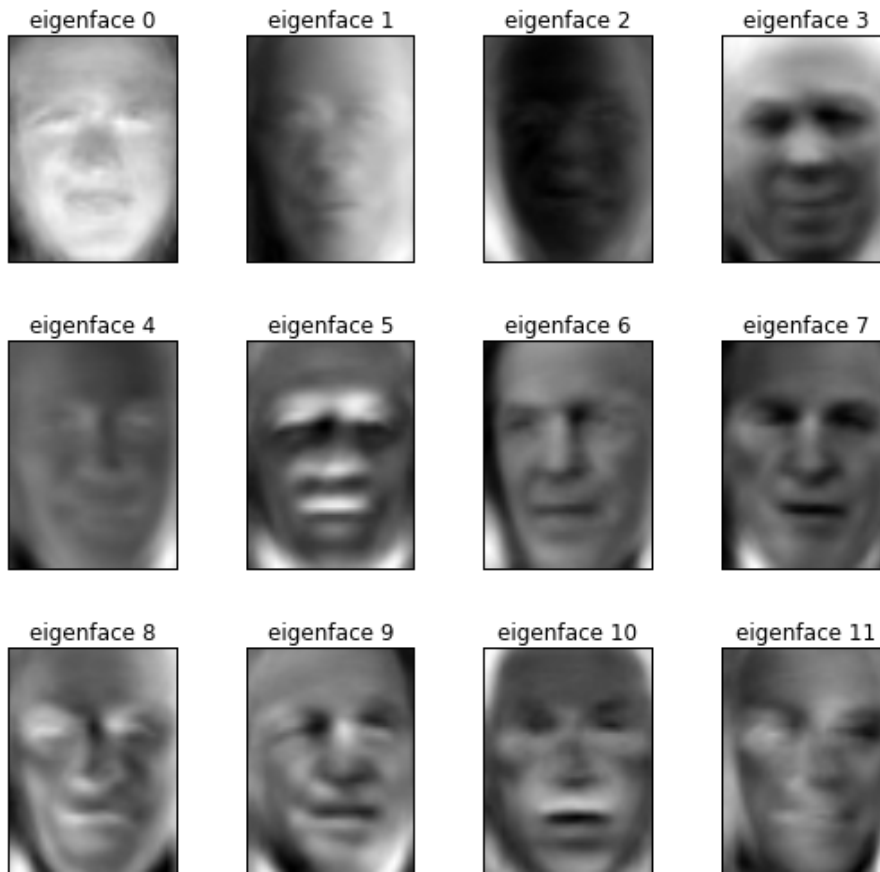
1  # A helper function to make plots of the faces
2  def plot_gallery(images, titles, h, w, n_row=3, n_col=4):
3      plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
4      plt.subplots_adjust(bottom=0, left=.01, right=.99,
5                          top=.90, hspace=.35)
6      for i in range(n_row * n_col):
7          plt.subplot(n_row, n_col, i + 1)

```

```

8 plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)
9 plt.title(titles[i], size=12)
10 plt.xticks(())
11 plt.yticks(())
12
13 # Plot the gallery of the most significant eigenfaces
14 eigenface_titles = [
15     "eigenface %d" % i for i in range(eigenfaces.shape[0])]
16
17 plot_gallery(eigenfaces, eigenface_titles, h, w)
18
19 plt.show()

```



([http://efavdb.com/wp-](http://efavdb.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-21-at-12.36.09-PM.png)

[content/uploads/2015/01/Screen-Shot-2015-01-21-at-12.36.09-PM.png](http://efavdb.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-21-at-12.36.09-PM.png))

Training a model

Now that we have reduced the dimensionality of the data it is time to go ahead and train a model. I am going to use the same SVM and GridSearchCV method I explained in my previous [post](http://efavdb.com/machine-learning-with-wearable-sensors/) (<http://efavdb.com/machine-learning-with-wearable-sensors/>). However, instead of using a linear kernel, as we did last time, I'll use instead a [radial basis function \(RBF\)](http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html) (http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html) kernel. The RBF kernel is a good choice here since we'd like to have non-linear decision boundaries — in general, it's a reasonable idea to try this out whenever the number of training examples outnumbers the number of features characterizing those examples. The parameter C here acts as a regularization term: Small C values give you smooth decision boundaries, while large C values

give complicated boundaries that attempt to fit/accommodate all training data. The gamma parameter defines how far the influence of a single point example extends (the width of the RBF kernel).

```

1  #Train a SVM classification model
2
3  print("Fitting the classifier to the training set")
4  t0 = time()
5  param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
6               'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }
7
8  clf = GridSearchCV(
9      SVC(kernel='rbf', class_weight='auto'), param_grid)
10 clf = clf.fit(X_train_pca, y_train)
11 print("done in %0.3fs" % (time() - t0))
12 print("Best estimator found by grid search:")
13 print(clf.best_estimator_)
14
15 >>
16 Fitting the classifier to the training set
17 done in 16.056s
18 Best estimator found by grid search:
19 SVC(C=1000.0, cache_size=200, class_weight='auto', coef0=0.0,
20     degree=3, gamma=0.001, kernel='rbf', max_iter=-1,
21     probability=False, random_state=None, shrinking=True,
22     tol=0.001, verbose=False)

```

Model validation

That's it for training! Next we'll validate our model on the testing data set. Below, we first use our PCA model to transform the testing data into our current feature space. Then, we apply our model to make predictions on this set. To get a feel for how well the model is doing, we print a [classification_report](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html) (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html) and a [confusion matrix](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html) (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html).

```

1  #Quantitative evaluation of the model quality on the test set
2  #Validate the data
3  X_test_pca = pca.transform(X_test)
4  y_pred = clf.predict(X_test_pca)
5
6  print(classification_report(
7      y_test, y_pred, target_names=target_names))
8
9  print('Confusion Matrix')
10 #Make a data frame so we can have some nice labels
11 cm = confusion_matrix(y_test, y_pred, labels=range(n_classes))
12 df = pd.DataFrame(cm, columns = target_names, index = target_names)
13 print(df)
14
15 >>
16
17
18
19

```

	precision	recall	f1-score	support
Ariel Sharon	0.81	0.85	0.83	20
Colin Powell	0.82	0.78	0.80	54

20	Donald Rumsfeld	0.78	0.67	0.72	27
21	George W Bush	0.87	0.95	0.91	139
22	Gerhard Schroeder	0.86	0.73	0.79	26
23	Hugo Chavez	1.00	0.75	0.86	20
24	Tony Blair	0.84	0.89	0.86	36
25					
26	avg / total	0.85	0.85	0.85	322
27					
28	Confusion Matrix				
29		Ariel Sharon	Colin Powell	Donald Rumsfeld	George W
30	Ariel Sharon	17	3	0	
31	Colin Powell	1	42	1	
32	Donald Rumsfeld	3	1	18	
33	George W Bush	0	3	3	
34	Gerhard Schroeder	0	1	1	
35	Hugo Chavez	0	0	0	
36	Tony Blair	0	1	0	
37					
38		Gerhard Schroeder	Hugo Chavez	Tony Blair	
39	Ariel Sharon	0	0	0	
40	Colin Powell	1	0	2	
41	Donald Rumsfeld	1	0	1	
42	George W Bush	0	0	1	
43	Gerhard Schroeder	19	0	2	
44	Hugo Chavez	1	15	0	
45	Tony Blair	0	0	32	

As a quick reminder, lets define what the terms above are:

1. precision is the ratio $Tp / (Tp + Fp)$ where Tp is the number of true positives and Fp the number of false positives.
2. recall is the ration of $Tp / (Tp + Fn)$ where Fn is the number of false negatives.
3. f1-score is $(precision * recall) / (precision + recall)$
4. support is the total number of occurrences of each face.

In our second table here, we have printed a confusion matrix, which provides a nice summary visualization of our results: Each row is the actual class, and the columns are the predicted class. For example, in row 1 there are 17 correct identifications of Arial Sharon, and 5 wrong ones. Using our previously defined helper plotting function, we show some examples of predicted vs true names below. Our simple algorithm's accuracy is imperfect, yet satisfying!

```

1 #Plot predictions on a portion of the test set
2 def title(y_pred, y_test, target_names, i):
3     pred_name = target_names[y_pred[i]].rsplit(' ', 1)[-1]
4     true_name = target_names[y_test[i]].rsplit(' ', 1)[-1]
5     return 'predicted: %s\ntrue: %s'%(pred_name, true_name)
6
7 prediction_titles = [title(y_pred, y_test, target_names, i)
8                      for i in range(y_pred.shape[0])]
9
10 plot_gallery(X_test, prediction_titles, h, w, 6, 4)
```


predicted: Powell
true: Powell



predicted: Bush
true: Bush



predicted: Chavez
true: Chavez



predicted: Bush
true: Bush



predicted: Rumsfeld
true: Rumsfeld



predicted: Blair
true: Schroeder



predicted: Sharon
true: Sharon



predicted: Schroeder
true: Schroeder



predicted: Powell
true: Powell



predicted: Bush
true: Bush



predicted: Blair
true: Powell



predicted: Blair
true: Blair



predicted: Bush
true: Bush



predicted: Blair
true: Schroeder



predicted: Sharon
true: Sharon



predicted: Bush
true: Bush



predicted: Bush
true: Bush



predicted: Powell
true: Powell



predicted: Powell
true: Powell



predicted: Powell
true: Powell



predicted: Powell
true: Bush



predicted: Blair
true: Blair



predicted: Blair
true: Blair



predicted: Rumsfeld
true: Rumsfeld



(<http://efavdb.com/wp-content/uploads/2015/01/download.png>)

Discussion

85% average accuracy shows that PCA (Eigenface analysis) can provide accurate face recognition results, given just a modest amount of training data. There are pros and cons to eigenfaces however:

- Pros
 1. Training can be automated.
 2. Once the the eigenfaces are calculated, face recognition can be performed in real time.
 3. Eigenfaces can handle large databases.
- Cons
 1. Sensitive to lighting conditions.
 2. Expression changes are not handled well.
 3. Has trouble when the face angle changes.
 4. Difficult to interpret eigenfaces: Eg, one can't easily read off from these eye separation distance, etc.

There are more advanced facial recognition methods that take advantage of features special to faces. One example is provided by the [Active Appearance Model \(AAM\)](http://en.wikipedia.org/wiki/Active_appearance_model) (http://en.wikipedia.org/wiki/Active_appearance_model), which finds facial features (nose, mouth, etc.), and then identifies relationships between these to carry out identifications. Whatever the approach, the overall methodology is the same for all facial recognition algorithms:

1. Take a labeled set of faces.
2. Extract features from those faces using some method of choice (eg eigenfaces).
3. Train a machine learning model on those features.
4. Extract features from a new face, and predict the identity.

The story doesn't end with finding faces in photos. Facial recognition is just a subset of machine vision, which is currently being applied widely in industry. For example, Intel and other semiconductor manufactures use machine vision to detect defects in the chips being produced — one application where by-hand (human) analysis is not possible and computers have the upper hand.

(http
 ://tw
 itter.
 com (http
 /inte ://re
 nt/t ddit.
 weet com
 ? /sub
 text mit?
 =Ma url=
 chin http:
 e //efa
 lear (http vdb.
 ning s://p com
 for lus.g /ma
 facia oogl chin
 l e.co e-
 reco m/s lear
 gniti hare ning
 on& ? -for-
 url= url= facia
 http: http: l-
 //efa //efa reco
 vdb. vdb. gniti
 com com on-
 /ma /ma 3/&ti
 chin chin tle=
 e- e- Mac
 lear lear hine
 ning ning lear
 -for- -for- ning
 facia facia for
 l- l- facia
 reco reco l
 gniti gniti reco
 on- on- gniti
 3/) 3/) on)



You're done! To see comments from people you follow, go to your Home feed on Disqus.

[See Home](#)[Dismiss](#) ✕**0 Comments****EFavDB** **Steven Vo** ▾ **Recommend** 1 **Share****Sort by Best** ▾

Start the discussion...

Be the first to comment.

ALSO ON EFavDB**WHAT'S THIS?**

Forecasting Bike Sharing Demand

4 comments • 5 months ago

Machine learning to predict San Francisco crime

1 comment • 12 days ago

 **Subscribe** **Add Disqus to your site** **Privacy**

◀ [Quick tutorial on MySQL](http://efavdb.com/quick-tutorial-on-mysql/) (http://efavdb.com/quick-tutorial-on-mysql/)

[NBA week 11 summary, week 12 predictions](http://efavdb.com/nba-week-11-summary-week-12-predictions/) ▶ (http://efavdb.com/nba-week-11-summary-week-12-predictions/)



WELCOME

EFavDB: Everybody's FAVORite Data Blog. Join us as we explore machine learning and data science!

FOLLOW US



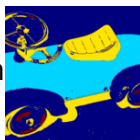
(http://
ps://
twitt
er.c
om/
efav
db)

POPULAR POSTS



Forecasting Bike Sharing Demand (<http://efavdb.com/bike-share-forecasting/>)

(<http://efavdb.com/bike-share-forecasting/>)



The mean shift clustering algorithm (<http://efavdb.com/mean-shift/>)

(<http://efavdb.com/mean-shift/>)



Machine learning for facial recognition (<http://efavdb.com/machine-learning-for-facial-recognition-3/>)

(<http://efavdb.com/machine-learning-for-facial-recognition-3/>)

avdb.com
/machine

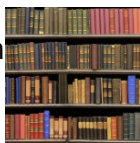


How not to sort by average rating, revisited

(<http://efavdb.com/ranking-revisited/>)

-learning-
for-facial-
recogniti
on-3/)

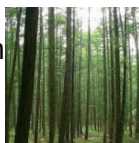
(<http://efavdb.com/ranking-revisited/>)



Quick tutorial on MySQL (<http://efavdb.com/quick-tutorial-on-mysql/>)

(<http://efavdb.com/quick-tutorial-on-mysql/>)

avdb.com
/quick-
tutorial-




Machine Learning Methods: Decision trees and forests

(<http://efavdb.com/notes-on-trees/>)

(<http://efavdb.com/notes-on-trees/>)

on-
mysql/)

avdb.com
/notes-
on-trees/)



Machine Learning Methods:
Classification without
negative examples
([http://efavdb.com/methods-
regression-without-negative-
avdb.comexamples/](http://efavdb.com/methods-regression-without-negative-examples/))
/methods
-
regressio
n-
without-
negative-
examples
/)

CATEGORIES

NBA prediction project (http://efavdb.com/category/nba-predict-project/)	26
MLB prediction project (http://efavdb.com/category/mlb-prediction-project/)	8
Case studies (http://efavdb.com/category/case-studies/)	7
Traffic project (http://efavdb.com/category/traffic-project/)	6
Methods (http://efavdb.com/category/methods-2/)	5
Guest posts (http://efavdb.com/category/guest-posts/)	3
Tools (http://efavdb.com/category/tools/)	2
Theory (http://efavdb.com/category/theory/)	2
Visualization (http://efavdb.com/category/visualization/)	1
Programming (http://efavdb.com/category/programming/)	1
Review (http://efavdb.com/category/review/)	1

RECENT POSTS

Machine learning to predict San Francisco crime (<http://efavdb.com/predicting-san-francisco-crimes/>)

How not to sort by average rating, revisited (<http://efavdb.com/ranking-revisited/>)

A review of the online course "Introduction to Big Data with Apache Spark" (<http://efavdb.com/review-intro-to-big-data-with-spark/>)

Multivariate Cramer-Rao inequality (<http://efavdb.com/multivariate-cramer-rao-bound/>)

Reshaping Data in R (<http://efavdb.com/reshaping-data-in-r/>)

TAGS

cloud-computing (<http://efavdb.com/tag/cloud-computing/>)

Databricks (<http://efavdb.com/tag/databricks/>) edX (<http://efavdb.com/tag/edx/>)

guest (<http://efavdb.com/tag/guest/>)

Machine Learning technique (<http://efavdb.com/tag/machine-learning-technique/>)

methods (<http://efavdb.com/tag/methods/>) mooc (<http://efavdb.com/tag/mooc/>)

NBA (<http://efavdb.com/tag/nba/>) programming (<http://efavdb.com/tag/programming/>)

R (<http://efavdb.com/tag/r/>) review (<http://efavdb.com/tag/review/>)

spark (<http://efavdb.com/tag/spark/>) statistics (<http://efavdb.com/tag/statistics/>)

traffic (<http://efavdb.com/tag/traffic/>) visualization (<http://efavdb.com/tag/visualization-2/>)

ARCHIVES

Select Month 

TAGS

cloud-computing (<http://efavdb.com/tag/cloud-computing/>) Databricks (<http://efavdb.com/tag/databricks/>)

[edx](http://efavdb.com/tag/edx/) (<http://efavdb.com/tag/edx/>) [guest](http://efavdb.com/tag/guest/) (<http://efavdb.com/tag/guest/>)

[Machine Learning technique](http://efavdb.com/tag/machine-learning-technique/) (<http://efavdb.com/tag/machine-learning-technique/>)

[methods](http://efavdb.com/tag/methods/) (<http://efavdb.com/tag/methods/>) [mooc](http://efavdb.com/tag/mooc/) (<http://efavdb.com/tag/mooc/>)

[NBA](http://efavdb.com/tag/nba/) (<http://efavdb.com/tag/nba/>) [programming](http://efavdb.com/tag/programming/) (<http://efavdb.com/tag/programming/>)

[R](http://efavdb.com/tag/r/) (<http://efavdb.com/tag/r/>) [review](http://efavdb.com/tag/review/) (<http://efavdb.com/tag/review/>) [spark](http://efavdb.com/tag/spark/) (<http://efavdb.com/tag/spark/>)

[statistics](http://efavdb.com/tag/statistics/) (<http://efavdb.com/tag/statistics/>) [traffic](http://efavdb.com/tag/traffic/) (<http://efavdb.com/tag/traffic/>)

[visualization](http://efavdb.com/tag/visualization-2/) (<http://efavdb.com/tag/visualization-2/>)

CATEGORIES

[NBA prediction project](http://efavdb.com/category/nba-predict-project/) (<http://efavdb.com/category/nba-predict-project/>)

[MLB prediction project](http://efavdb.com/category/mlb-prediction-project/) (<http://efavdb.com/category/mlb-prediction-project/>)

[Case studies](http://efavdb.com/category/case-studies/) (<http://efavdb.com/category/case-studies/>)

[Traffic project](http://efavdb.com/category/traffic-project/) (<http://efavdb.com/category/traffic-project/>)

[Methods](http://efavdb.com/category/methods-2/) (<http://efavdb.com/category/methods-2/>)

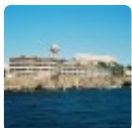
[Guest posts](http://efavdb.com/category/guest-posts/) (<http://efavdb.com/category/guest-posts/>)

POPULAR POSTS



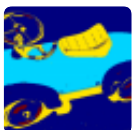
[Forecasting Bike Sharing Demand](http://efavdb.com/bike-share-forecasting/) (<http://efavdb.com/bike-share-forecasting/>)

26 Mar , 2015



[Machine learning to predict San Francisco crime](http://efavdb.com/predicting-san-francisco-crimes/) (<http://efavdb.com/predicting-san-francisco-crimes/>)

20 Jul , 2015



[The mean shift clustering algorithm](http://efavdb.com/mean-shift/) (<http://efavdb.com/mean-shift/>)

21 Apr , 2015

[NBA week 22 results, NBA season review, MLB week 1](http://efavdb.com/nba-week-22-results-nba-season-review-mlb-week-1/) (<http://efavdb.com/nba-week-22-results-nba-season-review-mlb-week-1/>)

19 Apr , 2015

ARCHIVES

Select Month



(http (http

:// ps:// ps://

:// twitt gith

ub.c er.c ub.c

om/ om/

efav efav

db) db)

[HOME \(HTTP://EFAVDB.COM/\)](http://efavdb.com/)

[ABOUT \(HTTP://EFAVDB.COM/ABOUT/\)](http://efavdb.com/about/)

[BAY AREA TRAFFIC \(HTTP://EFAVDB.COM/HBATA/\)](http://efavdb.com/hbata/)

[MLB DASHBOARD \(HTTP://EFAVDB.COM/MLB-DASH/\)](http://efavdb.com/mlb-dash/)

[WEEKLY MLB PREDICTIONS \(HTTP://EFAVDB.COM/WEEKLY-MLB-PREDICTIONS/\)](http://efavdb.com/weekly-mlb-predictions/)

EFavDB (<http://efavdb.com/>) All rights reserved. Theme by [Colorlib \(http://colorlib.com/\)](http://colorlib.com/) Powered by [WordPress \(http://wordpress.org/\)](http://wordpress.org/)