Cheatsheet – Python & R codes for common Machine Learning Algorithms

Analytics Vidhya — September 14, 2015

Beginner Business Analytics Cheatsheet Infographics Machine Learning Python R

Introduction

In his famous book – Think and Grow Rich, Napolean Hill narrates story of Darby, who after digging for a gold vein for a few years walks away from it when he was three feet away from it.

Now, I don't know whether the story is true or false. But, I surely know of a few Data Darby around me. These people understand the purpose of machine learning, its execution and use just a set 2 – 3 algorithms on whatever problem they are working on. They don't update themselves with better algorithms or techniques, because they are too tough or they are time consuming.

Like Darby, they are surely missing from a lot of action after reaching this close! In the end, they give up on machine learning by saying it is very computation heavy or it is very difficult or I can't improve my models above a threshold – what's the point? Have you heard them?

Today's cheat sheet aims to change a few Data Darby's to machine learning advocates. Here's a collection of 10 most commonly used machine learning algorithms with their codes in Python and R. Considering the rising usage of machine learning in building models, this cheat sheet is good to act as a code guide to help you bring these machine learning algorithms to use. Good Luck!

For the super lazy Data Darbies, we will make your life even easier. You can download the PDF Version of the cheat sheet here and copy paste the codes from it directly.

CHEATSHEET

Machine Learning Algorithms

(Python and R Codes)

Types

Supervised Learning

- Decision Tree
- Random Forest
- kNN
- Logistic Regression

Unsupervised Learning

- Apriori algorithm
- k-means
- Hierarchical Clustering

Reinforcement Learning

- Markov Decision Process
- Q Learning

Python Code

R Code

Linear Regression

```
#Import Library
#Import other necessary libraries like pandas,
#numpy...
from sklearn import linear_model
#Load Train and Test datasets
#Identify feature and response variable(s) and
#values must be numeric and numpy arrays
x_train=input_variables_values_training_datasets
y_train=target_variables_values_training_datasets
x_test=input_variables_values_test_datasets
#Create linear regression object
linear = linear_model.LinearRegression()
#Train the model using the training sets and
#check score
linear.fit(x_train, y_train)
linear.score(x_train, y_train)
#Equation coefficient and Intercept
print('Coefficient: \n', linear.coef_)
print('Intercept: \n', linear.intercept_)
#Predict Output
predicted= linear.predict(x_test)
```

```
#Load Train and Test datasets
#Identify feature and response variable(s) and
#values must be numeric and numpy arrays
x_train <- input_variables_values_training_datasets
y_train <- target_variables_values_training_datasets
x_test <- input_variables_values_test_datasets
x <- cbind(x_train,y_train)
#Train the model using the training sets and
#check score
linear <- lm(y_train ~ ., data = x)
summary(linear)
#Predict Output
predicted= predict(linear,x_test)
```

Logistic Regression

```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target)
#for training data set and x_test(predictor)
#of test_dataset
#Create logistic regression object
model = LogisticRegression()
#Train the model using the training sets
```

```
x <- cbind(x_train,y_train)
#Train the model using the training sets and check
#score
logistic <- glm(y_train ~ ., data = x,family='binomial')
summary(logistic)
#Predict Output
predicted= predict(logistic,x_test)
```

Log Regre

```
#and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

Decision Tree

```
#Import Library
#Import other necessary libraries like pandas, numpy...
from sklearn import tree
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of
#test_dataset
#Create tree object
model = tree.DecisionTreeClassifier(criterion='gini')
#for classification, here you can change the
#algorithm as gini or entropy (information gain) by
#default it is gini
#model = tree.DecisionTreeRegressor() for
#regression
#Train the model using the training sets and check
#score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(rpart)
x <- cbind(x_train,y_train)
#grow tree
fit <- rpart(y_train ~ ., data = x,method="class")
summary(fit)
#Predict Output
predicted= predict(fit,x_test)
```

SVM (Support Vector Machine)

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of test_dataset
#Create SVM classification object
model = svm.svc()
#there are various options associated
with it, this is simple for classification.
#Train the model using the training sets and check
#score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(e1071)
x <- cbind(x_train,y_train)
#Fitting model
fit <-svm(y_train ~ ., data = x)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)
```

Naive Bayes

```
#Import Library
from sklearn.naive_bayes import GaussianNB
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of test_dataset
#Create SVM classification object model = GaussianNB()
#there is other distribution for multinomial classes
like Bernoulli Naive Bayes
#Train the model using the training sets and check
#score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(e1071)
x <- cbind(x_train,y_train)
#Fitting model
fit <-naiveBayes(y_train ~ ., data = x)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)
```

kNN (k- Nearest Neighbors)

```
#Import Library
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of test_dataset
#Create KNeighbors classifier object model
KNeighborsClassifier(n_neighbors=6)
#default value for n_neighbors is 5
#Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(knn)
x <- cbind(x_train,y_train)
#Fitting model
fit <-knn(y_train ~ ., data = x,k=5)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)
```

k-Means

```
#Import Library
from sklearn.cluster import KMeans
#Assumed you have, X (attributes) for training data set
#and x_test(attributes) of test_dataset
#Create KNeighbors classifier object model
k_means = KMeans(n_clusters=3, random_state=0)
#Train the model using the training sets and check score
model.fit(X)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(cluster)
fit <- kmeans(X, 3)
#5 cluster solution
```

Random Forest

```
#Import Library
from sklearn.ensemble import RandomForestClassifier
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of test_dataset
#Create Random Forest object
model= RandomForestClassifier()
#Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Import Library
library(randomForest)
x <- cbind(x_train,y_train)
#Fitting model
fit <- randomForest(Species ~ ., x,ntree=500)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)
```

Dimensionality Reduction Algorithms

```
#Import Library
from sklearn import decomposition
#Assumed you have training and test data set as train and
#test
#Create PCA object pca= decomposition.PCA(n_components=k)
#default value of k =min(n_sample, n_features)
#For Factor analysis
#fa= decomposition.FactorAnalysis()
#Reduced the dimension of training dataset using PCA
train_reduced = pca.fit_transform(train)
#Reduced the dimension of test dataset
test_reduced = pca.transform(test)
```

```
#Import Library
library(stats)
pca <- princomp(train, cor = TRUE)
train_reduced  <- predict(pca,train)
test_reduced  <- predict(pca,test)
```

AdaBoost

```
#Import Library
from sklearn.ensemble import GradientBoostingClassifier
#Assumed you have, X (predictor) and Y (target) for
#training data set and x_test(predictor) of test_dataset
```

```
#Import Library
library(caret)
x <- cbind(x_train,y_train)
```

Gradient Boosting &

```
#training data set and x_test(predictor) of test_dataset
#Create Gradient Boosting Classifier object
model= GradientBoostingClassifier(n_estimators=100, \
         learning_rate=1.0, max_depth=1, random_state=0)
#Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

```
#Fitting model
fitControl <- trainControl( method = "repeatedcv",
+ number = 4, repeats = 4)
fit <- train(y ~ ., data = x, method = "gbm",
+ trControl = fitControl,verbose = FALSE)
predicted= predict(fit,x_test,type= "prob")[,2]
```


Keep this cheat sheet handy when you work on data sets. download the complete cheat sheet here: PDF Version

If you like what you just read & want to continue your analytics learning, subscribe to our emails, follow us on twitter or join our Facebook Group

decision tree infographic K-Means linear regression logistic regression machine learning Naive Bayes

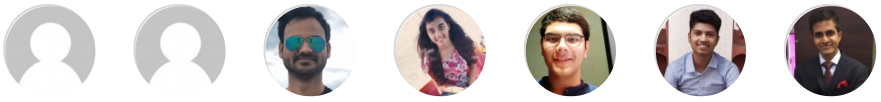
python R random forest Reinforcement Learning Supervised Learning unsupervised learning

About the Author

Analytics Vidhya

This is the official account of the Analytics Vidhya team.

Our Top Authors



Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Learn Gradient Boosting Algorithm for better predictions (with codes in R)

Next Post

24 Ultimate Data Scientists To Follow in the World Today

34 thoughts on "Cheatsheet – Python & R codes for common Machine Learning Algorithms"

venugopal says:
September 15, 2015 at 4:37 am

Good Compilation...

Reply

Indu says:
September 15, 2015 at 2:11 pm

Thanks for sharing this in both R and Python. Very helpful. It would be nice to have datasets to accompany this code for those who are just starting out....

Reply

Huaixiu Zheng says:
September 17, 2015 at 4:27 pm

thanks for sharing

Reply

Richard Boire says:
September 17, 2015 at 7:55 pm

This is good stuff. My only commentary is the following: I could not find anything in the code that deals with validation. This is a must in all models and their evaluation(i.e. how well the model performs in a holdout group). Evaluating the model based on its predicted output to observed output in the training data can be misleading because certain techniques have a tendency to overfit(i.e. neural nets) where holdout groups are essential in effectively evaluating model performance .

Reply

Kayla says:
September 18, 2015 at 1:08 pm

Awesome! Thank you!

Reply

---

greg says:
September 21, 2015 at 2:49 pm

thanks

Reply

---

andun says:
September 23, 2015 at 1:14 am

For kNN in R, the package knn is no longer available. The function knn can be found in the "class" package, but I don't think it takes the arguments the way you specified. I guess you can also use caret for that.

Reply

Gaurav says:
September 24, 2015 at 3:52 pm

Thx for sharing. In R we can implement stepwise regression.whats the equivalent in python.

Reply

Raman says:
October 12, 2015 at 3:42 am

Very thoughtfully compiled and presented. Thanks for posting something very useful!

Reply

Jared says:
November 13, 2015 at 3:32 pm

The download link has been invalid for China's mainland users. I tried to register the website but filed. Can someone please send the PDF file to my email ?

Reply

Analytics Vidhya Content Team says:
November 14, 2015 at 4:18 am

I've shared this PDF on your email.

Reply

venugopal says:
November 28, 2015 at 6:51 am

GooD one. Please share PDF file to my mail as well

Reply

Shikhar Pandey says:
November 29, 2015 at 10:58 am

Please share the PDF with me

Reply

Analytics Vidhya Content Team says:
November 30, 2015 at 6:05 am

Hi Venugopal Link to download is shared in the post above. You can very well download using the link. Thanks

Reply

socialin says:
December 28, 2015 at 1:30 am

"The reCAPTCHA wasn't entered correctly. Go back and try it again. (reCAPTCHA said: incorrect-captcha-sol)" The above messages prompted when I tried to register to download the pdf,I looked it up,it's about verify something,but I can't find the verify part anywhere.so that means I can't download it.Anyone know what's going on?

Reply

socialin says:
December 28, 2015 at 1:31 am

Can i get it in my email? Thanks!

Reply

Analytics Vidhya Content Team says:
December 28, 2015 at 4:13 am

Please share your email.

Reply

socialin says:
December 28, 2015 at 6:51 am

socialin@163.com,Thanks!

Reply

deepa says:
February 01, 2016 at 7:35 am

thank you

Reply

Manya says:
February 10, 2016 at 4:31 pm

Thank you

Reply

rajanikanth says:
February 11, 2016 at 9:13 am

pl fix the download link

Reply

dieudonne says:
February 22, 2016 at 1:21 am

can i get a copy by my email? thank you . guangfeizhao@outlook.com

Reply

Guilherme Cadori says:
March 10, 2016 at 3:59 pm

Hey, Manish. Could you please share it with me as well? email: gccadori@hotmail.com For some reason the link is not available. Cheers,

Reply

---

mobile live porn cam says:

June 09, 2016 at 10:10 pm

Can I simply just say what a comfort to uncover somebody that actually understands what they are discussing on the net. You certainly understand how to bring an issue to light and make it important. More people ought to read this and understand this side of your story. I was surprised that you aren't more popular given that you surely possess the gift.

Reply

Tikbal says:

June 15, 2016 at 1:36 am

there is a problem with the pdf link

Reply

Juan Pablo Garicoïts says:

July 17, 2016 at 8:59 pm

Nice compilation

Reply

---

gelou88 says:

August 23, 2016 at 9:35 pm

just to the point

Reply

---

stephen says:

August 26, 2016 at 5:32 am

Please share it with me thanks.

Reply

Manoj says:

August 26, 2016 at 10:31 am

Thanks for sharing. It's great help.

Reply

Vighneshwar Eligeti says:

September 18, 2016 at 4:33 pm

can i get a copy by my email? thank you . eligeti.vighneshwar.11ee1041@gmail.com

Reply

Prateek Tandon says:

October 25, 2016 at 5:28 pm

Good work. For code snippets though, using gists is more friendly to make updates as per updates to libraries etc.

Reply

Muhammad Fahmi Adli says:

November 07, 2016 at 1:11 am

Can I get that copy code by email? Thank you. My email is fahmi.ad26@gmail.com.

Reply

Brenda says:

November 10, 2016 at 2:21 am

please share the pdf to my email Thank you. God bless.

Reply

Harsh says:

January 04, 2017 at 10:48 am

Excellent piece of information. Posting both R and Python code is helpful in choosing which one to use for ML. Cheers! and Keep up the good work!

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

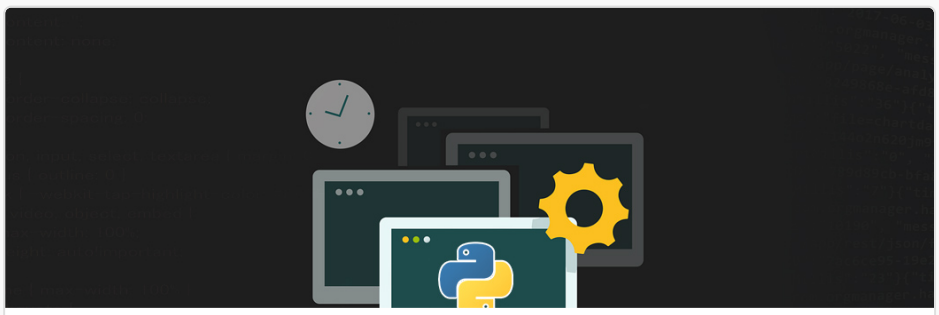
Name*

Email*

Website

Submit

Top Resources

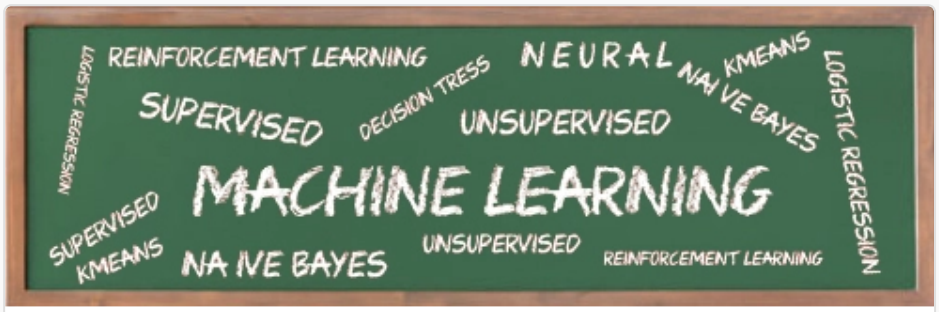


[Basic Concepts of Object-Oriented Programming in Python]

Himanshi Singh - SEP 01, 2020

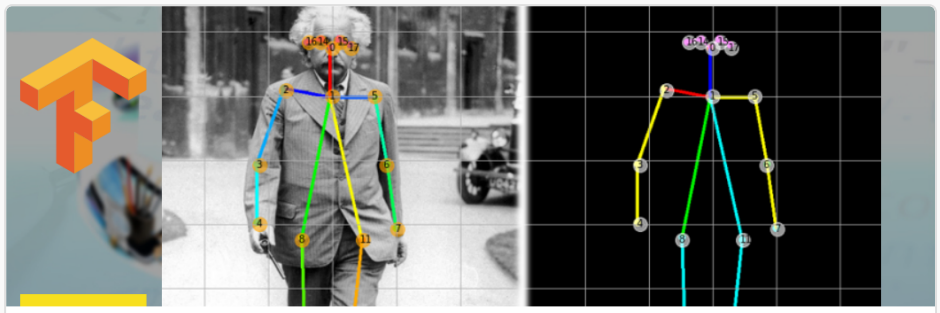
[Predicting Stock Prices with TensorFlow]

shivani46 - SEP 04, 2021



[Commonly used Machine Learning Algorithms (with Python and R Codes)]

Sunil Ray - SEP 09, 2017



[Posture Detection using PoseNet with Real-time Deep Learning project]

agrawal@71 - SEP 05, 2021