Operators

Arithmetic	(Addition(+), Substraction(-), Multiplication(*), Division(/), Modulus(%)
Relational	<, >, <=, >=, ==, != (not equal),
Assignment	=. +=, -=, /=, *=, %=
Logical	and. or, not
Membership	in, not in
Identity (same memory location)	is, is not

Functions

len()	determine the length of a string, a list, an array	
split()	split a string into shorter string using defined seperatos	string.split(“,”)

Functions (cont)

sum(),-mean(), count(), std().	functions that can be used by grouby in pandas	grouped_multiple = df.groupby(['Team', 'Pos']).agg({'Age': ['mean', 'min', 'max']}) grouped_multiple.columns = ['age_-mean', 'age_min', 'age_max'] grouped_multiple = grouped_multiple.reset_index()
		df.groupby(["Team","College"])["Salary"].max()
agg()	Allows for multiple or custom aggregations	defpct30(column):
		return column.quantile(0.3)
		dogs["weight_-kg"].agg(pct30)

Functions (cont)

keys()	We can use the Keys function of a Group By object to describe how rows of a dataset has been split	data.groupby(['month']).groups.keys()
join() and ravel()	An effective way to rename columns after a group	grouped = data.groupby('month').agg("duration": [min, max, mean])
		grouped.columns = ["_".join(x) for x in grouped.columns.ravel()]

Custom Functions

```
User-Defined Functions
By adding * to a parameter, we can add any number of arguments to that parameters
def func_w ith _va r_p os_ arg - s(* args):
for arg in args:
print(arg)
Simiarly, by adding * to an argument, we can add any number of arguments to that parameters
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Custom Functions (cont)

```
> def func_with_var_pos_args(*args):
for arg in args:
print(arg)
```

Naming Conventions

Funciton	function, my_function
Variable	x, var, my_variable
Class	Model, MyClass
Method	class_method, method

Packaging and Displaying

from pprint import pprint	pprint(dir(my_dict))
Pychecker	detects bugs from the source code and warns about its style and complexity
Pylint	Checks whether the module matches upto a coding standard.
Modules	Each Python program file is a module, importing other attributes and objects.
Package	folder of modules

Map, Filter and Lambda

Map	Applies a function to the input list	map(function_to_apply, list_of_inputs)	items = [1, 2, 3, 4, 5] squared = list(map(lambda x: x**2, items))
filter	creates a list of elements for which a function returns true.	filter(function_to_apply,nlisst_to_select_From)	number_list = range(-5, 5) less_than_zero = list(filter(lambda x: x < 0, number_list))
Reduce	applies a rolling computation to sequential pairs of values in a list	from functools import reduce	product = reduce((lambda x, y: x * y), [1, 2, 3, 4])

Scikit Learn - Regression

```
poly_reg =
PolynomialFeatures(degree = 2)
X_poly = poly_r eg.f it _tr -
ans for m(x train)
X_poly.pr edi ct( xtest)
xtrainp= X_poly [:1 1900*3]
# polynomial regression model
poly_r eg_ model = Linear Reg -
res sion()
poly_r eg_ mod el.f it (xt -
rainp, ytrain)
poly_r eg_ mod el.p re dic t(x -
test)
print( metric s.m ean _sq uar -
ed_ err or( y_test, poly_r eg_ -
mod el.p re dic t(x test) ) )
svr_re gressor = SVR(ke rne -
l=' rbf', gamma= 'auto')
svr_re gre sso r.f it( xtrain,
ytrain)
tree_r egr essor = Decisi onT -
ree Reg res sor (ra ndo m_state
= 0)
tree_r egr ess or.f it (xt rain,
ytrain)
forest _re gressor = Random -
For est Reg res sor (n_ est -
imators = 300, random _state =
0)
forest _re gre sso r.f it( -
xtrain, ytrain)
from sklearn import linear -
_model
reg = linear _mo del.La sso -
Lar s(a lph a=.1, normal ize -
=False)
reg.fi t(x train, ytrain)
reg.coef_
reg.pr edi ct( xtest)
est = SGDCla ssi fier()
est.fi t(x train, ytrain)
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Scikit Learn - Regression (cont)

```
> est.predict(xtest)
linear_regression = LinearRegression()
y_pred_lr = linear_regression.fit(xtrain,
ytrain).predict(xtest)
xgbmodel = xgboost.XGBRegressor(col-
sample_bytree=0.4,
                gamma=0,
                learning_rate=0.07,
                max_depth=3,
                min_child_weight=1.5,
                n_estimators=10000,
                reg_alpha=0.75,
                reg_lambda=0.45,
                subsample=0.6,
                seed=42)
xgbmodel.fit(xtrain, ytrain)
print( svr_regressor.predict(xtest))
print( tree_regressor.predict(xtest))
print( y_pred_lr)
print( forest_regressor.predict(xtest))
model.predict(xtest)
print( metrics.mean_squared_error(y_test,
svr_regressor.predict(xtest) ) )
print( metrics.mean_squared_error(y_test,
tree_regressor.predict(xtest) ) )
```

Scikit Learn - Regression (cont)

```
> print( metrics.mean_squared_error(y_test,
y_pred_lr) )
print( metrics.mean_squared_error(y_test,
forest_regressor.predict(xtest) ) )
forestrev = forest_regressor.predict(xtest)
xgbmodel.predict(xtest).mean()
print( metrics.mean_squared_error(y_test,
xgbmodel.predict(xtest) ) )
ytest.mean()
bas.REVENUE.mean()
xtrain, ytrain = np.array(xtrain), np.array(-
ytrain)
xtrain = np.reshape(xtrain, (xtrain.shap-
e[0],xtrain.shape[1],1))
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_seque-
nces=True, input_shape=(xtrain.shap-
e[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))
ytrain = ytrain.astype(np.float32)
xtrain = xtrain.astype(np.float32)
xtrain = np.reshape(xtrain, (xtrain.shap-
e[0],xtrain.shape[1],1))
```

Looping Data Structures

```
1) With One Column:
import pandas as pd
#The column to look through
brics = pd.rea d_c sv( " bri -
cs.c sv ", index_col = 0)
       for val in brics :
                    pri nt(val)
2) Index then all cols in row:
for lab, row in brics.i te rro -
ws():
      pri nt(lab)
      pri nt(row)
3) Index then one col in row:
for lab, row in brics.i te rro -
ws():
        bri cs.l oc [lab, " -
nam e_l eng th"] = len(ro w["c -
oun try "])
4) Apply
brics[ " nam e_l eng th"] =
brics[ " cou ntr y"].a pp -
ly(len)
```

Scikit Learn - Classification

```
## Classifier imports
from sklear n.n eig hbors import
KNeigh bor sCl ass ifier
from sklear n.n aiv e_bayes
import Gaussi anNB, Multin omi -
alNB, Bernou lliNB
from sklear n.l ine ar_ model
import Logist icR egr ession,
SGDCla ssifier
from sklear n.svm import SVC,
LinearSVC, NuSVC
from sklear n.l ine ar_ model
import Ridge
from sklear n.e nsemble import
AdaBoo stC las sifier
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Scikit Learn - Classification (cont)

```
> from sklearn.ensemble import GradientB-
oostingClassifier
# Defining our models
gnb = GaussianNB()
KNN = KNeighborsClassifier(n_neighbo-
rs=1)
MNB = MultinomialNB()
BNB = BernoulliNB()
LR = LogisticRegression()
SDG = SGDClassifier()
#SVC = SVC(kernel='linear', C=1e3)
LSVC = LinearSVC()
NSVC = NuSVC()
# Train our classifier and print accuracy
scores
gnb.fit(x1, y1)
y2_GNB_model = gnb.predict(x2)
print("GaussianNB Accuracy :", accuracy_-
score(y2, y2_GNB_model))
KNN.fit(x1,y1)
y2_KNN_model = KNN.predict(x2)
print("KNN Accuracy :", accuracy_score(y2,
y2_KNN_model))
#MNB.fit(x1,y1)
#y2_MNB_model = MNB.predict(x2)
#print("MNB Accuracy :", accuracy_sco-
re(y2, y2_MNB_model))
BNB.fit(x1,y1)
```

Scikit Learn - Classification (cont)

```
> y2_BNB_model = BNB.predict(x2)
print("BNB Accuracy :", accuracy_score(y2,
y2_BNB_model))
LR.fit(x1,y1)
y2_LR_model = LR.predict(x2)
print("LR Accuracy :", accuracy_score(y2,
y2_LR_model))
SDG.fit(x1,y1)
y2_SDG_model = SDG.predict(x2)
print("SDG Accuracy :", accuracy_score(y2,
y2_SDG_model))
# SVC.fit(x1,y1)
# y2_SVC_model = SVC.predict(x2)
# print("SVC Accuracy :", accuracy_sco-
re(y2, y2_SVC_model))
LSVC.fit(x1,y1)
y2_LSVC_model = LSVC.predict(x2)
print("LSVC Accuracy :", accuracy_sco-
re(y2, y2_LSVC_model))
NSVC.fit(x1,y1)
y2_NSVC_model = NSVC.predict(x2)
print("NSVC Accuracy :", accuracy_sco-
re(y2, y2_NSVC_model))
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com