

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.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_with_var_pos_args(* args):
    for arg in args:
        print(arg)
```

Similarly, by adding * to an argument, we can add any number of arguments to that parameters

Custom Functions (cont)

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

Naming Conventions

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

Packaging and Displaying

```
from pprint import pprint
pprint(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)
```

filter creates a list of elements for which a function returns true.

```
filter(function_to_apply, nlist_to_select_From)
```

Reduce applies a rolling computation to sequential pairs of values in a list

```
from functools import reduce
reduce(lambda x, y: x * y, [1, 2, 3, 4])
```

Scikit Learn - Regression

```
poly_reg = PolynomialFeatures(degree = 2)
X_poly = poly_reg.fit_transform(x_train)
# polynomial regression model
poly_reg_model = LinearRegression()
poly_reg_model.fit(x_train, y_train)
poly_reg_model.predict(x_test)

print(metrics.mean_squared_error(y_test, poly_reg_model.predict(x_test)))

svr_regressor = SVR(kernel='rbf', gamma='auto')
svr_regressor.fit(x_train, y_train)

tree_regressor = DecisionTreeRegressor(random_state = 0)
tree_regressor.fit(x_train, y_train)

forest_regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
forest_regressor.fit(x_train, y_train)

from sklearn import linear_model
reg = linear_model.Lasso(alpha=.1, normalize=False)
reg.fit(x_train, y_train)
reg.coef_
reg.predict(x_test)

est = SGDClassifier()
est.fit(x_train, y_train)
```



By **datamansam**

Published 15th May, 2022.

Last updated 9th July, 2022.

Page 2 of 4.

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.read_csv( " bri -
cs.csv ", index_col = 0)
    for val in brics :
        print(val)
2) Index then all cols in row:
for lab, row in brics.iterrows():
    print(lab)
    print(row)
3) Index then one col in row:
for lab, row in brics.iterrows():
    brics.iloc[lab, " -
name_length"] = len(row["c -
ountry"])
4) Apply
brics[" name_length"] =
brics[" cou ntr y"].apply(
len)
```

Scikit Learn - Classification

```
## Classifier imports
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.naive_bayes
import GaussianNB, MultinomialNB,
BernoulliNB
from sklearn.linear_model
import LogisticRegression,
SGDClassifier
from sklearn.svm import SVC,
LinearSVC, NuSVC
from sklearn.linear_model
import Ridge
from sklearn.ensemble import
AdaBoostClassifier
```



By **datamansam**

Published 15th May, 2022.

Last updated 9th July, 2022.

Page 3 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Scikit Learn - Classification (cont)

```
> from sklearn.ensemble import GradientBoostingClassifier
# Defining our models
gnb = GaussianNB()
KNN = KNeighborsClassifier(n_neighbors=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_score(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_score(y2, y2_SVC_model))
LSVC.fit(x1,y1)
y2_L SVC_model = LSVC.predict(x2)
print("LSVC Accuracy :", accuracy_score(y2, y2_L SVC_model))
NSVC.fit(x1,y1)
y2_NSVC_model = NSVC.predict(x2)
print("NSVC Accuracy :", accuracy_score(y2, y2_NSVC_model))
```



By **datamansam**

cheatography.com/datamansam/

Published 15th May, 2022.

Last updated 9th July, 2022.

Page 4 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>