id	library/module	category	name	usage	remarks	url	備考
1	built-in	function	strip()	cats['type'] = cats['split'].map(lambda x: x[0].strip())	・strからスペースを除外する ・Return a copy of the string S with leading and trailing whitespace removed.		
2	pandas	method	pd.set_option()	ex) import pandas as pd pd.set_option('max_columns',100) pd.set_option('max_rows',100)		https://qiita.com/tanemaki/items/2e	

3	pandas	method	pd.read_csv()	ex)
				data01 = pd.read_csv('report1516361407832.csv',encoding="cp932", **dtype=np.float32**)
				data01.head(10)
				
				ex2)
				# 区切りに";"がついているので注意
				student_data_math = pd.read_csv('student-mat.csv', sep=';')
				
				ex3)
				#ネット上のcsv読み込み
				url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'
				wine_data = pd.read_csv(url, sep=";")
				wine_data.head()
				
				ex4)
				# encodingの指定、ヘッダー、区切り文字の指定
				text = pd.read_csv('http://www.cl.ecei.tohoku.ac.jp/nlp100/data/hightemp.txt', encoding='utf-8',
				header=None, sep='\t')
				text.columns = ['prefecture', 'point', 'temperature', 'date']
				
				# ヘッダーがない場合に、csv読み込みと同時に列名をつける場合----------
				# ファイル読み込み
				data = pd.read_csv('/content/data.txt', sep='\t', header=None, names=['prefecture', 'point'])
				
				# 1列目のデータの一意な値の抽出
				np.unique(data.loc[:, 'prefecture'])
					・読み込んだcsvはdataframeになっている。
					・デフォルトで64bitでデータ保持しているので、**dtype=np.float32**とするとメモリの節約になる		

4	re		compile()	ex) p = re.compile(r"<[^>]*?>")		https://qiita.com/kaeruair/items/747	
5	pandas	type	dataframe	ex) 列抽出 data = data[['column1', 'column2']] data.column1 #カンマ+列名を付け足すだけでも抽出できる 行抽出		https://qiita.com/koara-local/items/(	

id	library/module	category	name	usage	remarks	url	備考

6	pandas	method	pd.DataFrame()	ex)
				※Xはscipy.sparse.csr.csr_matrix型
				test = pd.DataFrame(X.toarray().transpose()) #.transpose()は転置
				test.head(10) #columnが文書
				
				ex2)
				#列名のある4列の空のDFを作成 (行数は0)
				name_class_df = pd.DataFrame(columns=['miss','mrs','master','mr'])
				
				ex3)
				#1行4列のDFを作成
				df = pd.DataFrame([[1,0,0,0]],**columns=**['miss','mrs','master','mr'])
				# mrs miss mr master
				# 0 1 0 0 0
				
				ex4)
				from pandas import Series, DataFrame
				attri_data1 = {'ID':['100','101','102','103','104'],
				'City':['Tokyo','Osaka','Kyoto','Hokkaido','Tokyo'],
				'Birth_year':[1990,1989,1992,1997,1982],
				'Name':['Hiroshi','Akiko','Yuki','Satoru','Steve']}
				
				attri_data_frame1 = DataFrame(attri_data1)
				
				print(attri_data_frame1)
				
				#indexを指定してデータフレームを作成する
				attri_data_frame_index1 = DataFrame(attri_data1,index=['a','b','c','d','e'])
				
				ex5)
				#区間[0, 1]上の一様分布に従う乱数を発生させる
				rand_num1 = np.random.uniform(0.0, 1.0, 10000)
				rand_num2 = np.random.uniform(0.0, 1.0, 10000)
				
				rand_df = pd.DataFrame({"X":rand_num1, "Y":rand_num2})
				
				ex6)階層型インデックス
				# 3列3行のデータを作成し、インデックスとカラムを設定
				hier_df= DataFrame(
				np.arange(9).reshape((3,3)),
				index = [
				['a','a','b'],
				[1,2,2]
				],
				columns = [
				['Osaka','Tokyo','Osaka'],
				['Blue','Red','Red']
				]
				)
				hier_df
				
				#辞書型もDFに変換できる
				# 閾値を0.01から0.99の間で50通りとして、偽陽性率と真陽性率を計算
				rates = {}
				for threshold in np.linspace(0.01, 0.99, num=50):
				labels = results['benign'].map(**lambda x: 1 if x > threshold else 0**)
				m = confusion_matrix(y_test, labels)
				rates[threshold] = {'false positive rate': m[0,1] / m[0, :].sum(),
				'true positive rate': m[1,1] / m[1, :].sum()}
				
				# 横軸をfalse positive rate、縦軸をtrue positive rateとしてプロット
				pd.DataFrame(rates).T.plot.scatter('false positive rate', 'true positive rate')
				
				# indexに名前を付ける
				hier_df.index.names =['key1','key2']
				# カラムに名前を付ける
				hier_df.columns.names =['city','color']
				hier_df
				
				# 階層ごとの要約統計量:行合計
				hier_df.sum(level = 'key2', axis = 0)
					・exはarrayからData.Frameに変換する例
					・注)columns=[]でなくcolumns={}として列名を指定すると、列名の並びが指定した順でなくなる
					・DataFrameオブジェクトは2次元の配列です。それぞれの列で、異なるdtype（データ型）を持たせることもできる
					・DataFrameオブジェクトもSeriesオブジェクトと同様にインデックスを変更したり、インデックスとして文字を指定したりすることもできます
					
					・dict型を引数にとってpd.DataFrame()する場合かつ、dictの各要素が1個しかない場合はindex = []として行名を決めないとエラーになる
					ValueError: If using all scalar values, you must pass an index
						https://note.nkmk.me/python-panda
						http://www.mirandora.com/?p=180	

id	library/module	category	name	usage	remarks	url	備考
7	pandas	method	.T	from pandas import Series, DataFrame attri_data1 = {'ID':['100','101','102','103','104'], 'City':['Tokyo','Osaka','Kyoto','Hokkaido','Tokyo'], 'Birth_year':[1990,1989,1992,1997,1982], 'Name':['Hiroshi','Akiko','Yuki','Satoru','Steve']} attri_data_frame1 = DataFrame(attri_data1) attri_data_frame1**.T**	・行列を転置する	松尾研	
8	pandas	accessor	str	ex) df.column1.str	dataframeに使えるアクセサで、文字列を取り出す。	https://qiita.com/knknkn1162/items	
9	pandas	method	pd.df.replace()	ex) df.replace(r'HAGE.', 'HAGEeeeee', regex=True) **ex2)#DF内の値(文字列)に対して関数を適用する場合** data["open"].str.replace("$", "") ex3) delete_dolchar = lambda x: str(x).replace("$", "")	・正規表現を使う際はregex=Trueのおまじないが必要。 ・文字列置換	http://nekoyukimmm.hatenablog.co	
10	selenium	webdriver	find_elements_by_css_selector()	ex) game_title = driver.find_elements_by_css_selector('#body-content > div > div > div.main-content > div > div > div > div.id-card-list.card-list.two-cards > div:nth-child(n) > div > div.details > a.title')	n番目の子を選択できる。:nth-child(1), :nth-child(2) などの指定も可能	http://css3.under.jp/selector/pseud	

11	pandas	method	DataFrame.to_csv()	to_csv(path_or_buf=None, sep=',', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None, tupleize_cols=None, date_format=None, doublequote=True, escapechar=None, decimal='.')
				
				ex1)
				# header=False→ヘッダー行なしで書き出し、index=False→行名なしで書き出し
				text.to_csv('text.txt', header=False, sep='\t', index=False)
					・method of pandas.core.frame.DataFrame instance
					Write object to a comma-separated values (csv) file.
					
					■引数
					・index
					行名をファイル出力するか否か
					Falseだと行名を出力しない
						https://pythondatascience.plavox.ir	

12	pandas	method				https://note.nkmk.me/python-panda	
13	pandas	method	DataFrame. to_pickle ('file_name.pkl')	ex) game_list_df.to_pickle('game_list_df.pkl') #読み込む際 #test = game_list_df.read_pickle('game_list_df.pkl')		https://qiita.com/glires/items/7338a	
14	pandas	method	DataFrame. read_pickle ('file_name.pkl')	ex) game_list_df.to_pickle('game_list_df.pkl') #読み込む際 #test = game_list_df.read_pickle('game_list_df.pkl')		https://qiita.com/glires/items/7338a	

15	matplotlib	method	hist()	hist(x, bins=None, range=None, density=None, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, normed=None, *, data=None, **kwargs)
				
				ex)
				import numpy as np
				import matplotlib.pyplot as plt
				%matplotlib inline
				
				x = np.random.normal(size = 100)
				
				plt.hist(x)
				plt.title("Histgram")
				plt.xlabel("x")
				plt.ylabel("frequency")
				plt.show()
				
				ex2)
				# シードの固定
				random.seed(0)
				
				# グラフの大きさ指定
				plt.figure(figsize = (20, 6))
				
				# ヒストグラムの描写
				plt.hist(np.random.randn(10 ** 5) * 10 + 50, **bins = 60**, **range = (20, 80)**)
				
				plt.grid(True)
					・bins→ビンの数(幅、個数)を指定
					・range→範囲を指定
					・「?plt.hist」と打つと、利用できるパラメタを確認できる
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

16	pandas.plotting._core	method	hist()	# データを読み込む
				student_data_math = pd.read_csv('student-mat.csv', sep=';')
				
				# カーネル密度関数
				student_data_math.absences.plot(kind='kde', style='k--')
				
				# 単純なヒストグラム、density=Trueにすることで、確率で表示
				student_data_math.absences.hist(density=True)
				plt.grid(True)			

17	pandas	アクセサ	df.columns	ex) test.columns = ['rank', 'title'] #1列だけの場合も、[]で囲って指定しないとエラーになる tmp = pd.DataFrame(close_data_corr_by_stock.unstack()) tmp.columns = ["corr_value"]	列名を変更する	https://note.nkmk.me/python-panda	
18	pandas.core.generic	method	df.describe()	describe(percentiles=None, include=None, exclude=None) ex) train.describe(include="O") #train.describe(include="object")と同義 **#小数点以下の桁数を指定する場合** train.describe().round(2) #小数点以下2桁までに指定 ex2) #列を絞ってdescribe student_data_math['absences'].describe() # 四分位範囲(75%タイル ー 25%タイル) student_data_math['absences'].describe()[6] - student_data_math['absences'].describe()[4]	・要約統計量を出す データ数、平均値、標準偏差、最小値、25、50、75パーセンタイル値、そして最大値を計算 ・describeメソッドの結果は、Seriesオブジェクトに入る それぞれの要素は、describe()[インデックス番号]として取得できます。たとえば、平均値を示すmeanの値は、describe()[1]、標準偏差を示すstdの値はdescribe()[2] ・include引数に要約対象のデータ型を指定できる ・include = "O", include="object"でカテゴリ型？を要約対象にできる ・include = "all"で全オブジェクト method of pandas.core.frame.DataFrame instance Generates descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding ``NaN`` values	https://www.sejuku.net/blog/40241 https://note.nkmk.me/python-panda	
19	sklearn.feature_extraction.text	関数	CountVectorizer	ex) from sklearn.feature_extraction.text import CountVectorizer vectorizer = **CountVectorizer(min_df=1, stop_words='english')**	sklearnのCountVectorizerを使うとBoW(Bag of Words)の特徴量が簡単に作れます。ただし、指定するパラメタが多かったり、デフォルトで英語の文字列を想定していたりして若干とっつきづらい部分もあります。 **・min_df=1** 頻繁に使われていない単語をCountVectorizerが無視する際に使う 設定する値が整数→その数より出現回数が小さい単語は無視される 設定する値が分数→データセットの数にその分数をかけた値を基準として、それより出現回数が小さい場合に無視する **max_df**も同様 **・stop_words="** ストップワード(重要度が低く、除外する単語)を指定できる englishと指定すると予めストップワードとして登録されている318個の単語をストップワードとして登録することができる	https://hayataka2049.hatenablog.jp	
20	sklearn.feature_extraction.text		TfidfVectorizer	ex) from sklearn.feature_extraction.text import TfidfVectorizer	TFIDF計算に使えるらしい		
21	sklearn.feature_extraction.text	method	get_stop_words()	ex) sorted(vectorizer.get_stop_words())[0:20]	どのような単語がストップワードとして登録されているか見る		
22	sklearn.feature_extraction.text	method	fit_transform (corpus)	ex) corpus = ['All my cats in a row', 'When my cat sits down, she looks like a Furby toy!', 'The cat from outer space', 'Sunshine loves to sit like this for some reason.'] vectorizer = CountVectorizer() features = vectorizer.**fit_transform(corpus)**.todense() print(vectorizer.vocabulary_)	Learn the vocabulary dictionary and return term-document matrix.	http://scikit-learn.org/stable/module	
23	sklearn.preprocessing.label	method	fit_transform()	ex) from sklearn.preprocessing import LabelEncoder shops['city_code'] = LabelEncoder().fit_transform(shops['city'])	・Fit label encoder and return encoded labels	https://scikit-learn.org/stable/module	
24	sklearn.metrics.pairwise		euclidean_distances	ex) from sklearn.metrics.pairwise import euclidean_distances	2点間のユークリッド距離を求める		
25	?	method	todense()	ex) features = vectorizer.fit_transform(corpus).todense()	numpy.ndarrayに変換する時はtoarray()、numpy.matrixに変換する時はtodense()を使います	https://ohke.hateblo.jp/entry/2018/(	

id	library/module	category	name	usage	remarks	url	備考

26	numpy	method	toarray()	ex)
				content = ["How to format my hard disc", "Hard disk format problems"]
				X = vectorizer.fit_transform(content) #BoWを作成 #scipy.sparse.csr.csr_matrix
				vectorizer.get_feature_names() #特徴量(トークン)を表示
				print(X.toarray().transpose())
					numpy.ndarrayに変換		

27	pandas	method	transpose()	ex)
				content = ["How to format my hard disc", "Hard disk format problems"]
				X = vectorizer.fit_transform(content) #BoWを作成 #scipy.sparse.csr.csr_matrix
				vectorizer.get_feature_names() #特徴量(トークン)を表示
				print(X.toarray().transpose())
					行列(配列)を転地させる	https://note.nkmk.me/python-panda	

28	base	method	sorted()	sorted(iterable, /, *, key=None, reverse=False) ex) >>> sorted([5, 2, 3, 1, 4]) [1, 2, 3, 4, 5]	単純な昇順のソートはとても簡単です: sorted() 関数を呼ぶだけです。そうすれば、新たにソートされたリストが返されます Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.	https://docs.python.jp/3/howto/sorti	
29	built-in	function	list()	list('cat') # ['c', 'a', 't']	・list()に文字列を入れると1文字ずつのリストを返す		
30	numpy	method	tolist()	ex) data = pd.read_csv("pokemon_go_comment.csv") corpus = data.text.values.tolist() #列名でDFからデータ抽出できる #pandas.core.series.Series → #list	pandas.DataFrame, pandas.Seriesをリスト型に直接変換するメソッドは無いため、values属性で取得できるNumPy配列ndarrayを経由して、ndarrayのtolist()メソッドでリストに変換する。	https://note.nkmk.me/python-panda	
31	pandas	method	http://kamonohashiperry.com/archives/1619 http://www.tohoho-web.com/python/operators.html	ex) nakamoto_test = nakamoto_corpus.sample(frac=0.1, replace=True) #pandas.core.frame.DataFrame 402行3列ほど	fracに指定した割合でデータセットからランダムにデータを抜き出す関数と思われる	http://kamonohashiperry.com/archi	
32	演算子	その他	~	nakamoto_train= nakamoto_corpus[~nakamoto_corpus.index.isin(nakamoto_test.index)] #pandas.core.frame.DataFrame	Rでいうところの！と同じだと思う。否定を表す	http://kamonohashiperry.com/archi http://www.tohoho-web.com/pythor	
33	演算子	その他	//	=a　　# 正数 -a　　# 負数 a + b　　# 加算 a - b　　# 減算 a * b　　# 乗算 a / b　　# 除算 a % b　　# a を b で割った余り a ** b　　# a の b 乗 a // b　　# 切り捨て除算		http://www.tohoho-web.com/pythor	

34	DataFrame in module pandas.core.frame	parameters	index	ex)
				nakamoto_corpus.index
				
				train.index = pd.to_datetime(train["datetime"]) #train.indexがpandasのdatetime型
				train.head()
					・RのDFで言うところのrownamesみたいなもの	http://kamonohashiperry.com/archi	

35	base	method	str.split()	str.split(sep=None, maxsplit=-1) ex) # 引数を指定しなければ、スペースで分割されます 'Guardians of the Galaxy'.split() # => ['Guardians', 'of', 'the', 'Galaxy'] #datetimeからmonthを抜き出す train["month"] = train["datetime"].apply(lambda x : int(x.split("-")[1]))	文字列を sep をデリミタ文字列として区切った単語のリストを返します	https://www.lifewithpython.com/20	

id	library/module	category	name	usage	remarks	url	備考
36	base	method	str.join()	l = ['aaa', 'bbb', 'ccc'] s = ''.join(l) print(s) # aaabbbccc s = ','.join(l) print(s) # aaa,bbb,ccc s = '-'.join(l) print(s) # aaa-bbb-ccc s = '\n'.join(l) print(s) # aaa # bbb # ccc	・splitの反対 ・**'間に挿入する文字列'.join([連結したい文字列のリスト])** ・リストの要素を結合して文字列にする時など	https://note.nkmk.me/python-string-	
37	base	method	append()	リストオブジェクト.append(オブジェクト) ex) list = ["A", "B", "C"] list.append("D") print list　# ["A", "B", "C", "D"] # リスト同士をつなげたい場合は以下も可能 ['abc', 'efg'] + ['hij'] # ['abc', 'efg', 'hij']	・リスト型で用意されているメソッド ・オブジェクトをリストの最後に追加する	https://www.pythonweb.jp/tutorial/li	

id	library/module	category	name	usage	remarks	url	備考

38	pandas	method	isin()	Series.isin(values)
				
				ex)
				>>> s = pd.Series(['lama', 'cow', 'lama', 'beetle', 'lama',
				... 'hippo'], name='animal')
				>>> s.isin(['cow', 'lama'])
				0 True
				1 True
				2 True
				3 False
				4 True
				5 False
				Name: animal, dtype: bool
				
				ex2)
				nakamoto_train= nakamoto_corpus[~nakamoto_corpus.index.isin(nakamoto_test.index)]
				#テストデータに含まれなかったものを訓練データとしている
				#~nakamoto_corpus.index.isin(nakamoto_test.index)が論理値のarrayになっている 論理値でDFの行指定してデータ抽出している
				
				ex3)
				#上位5社に絞る
				test2 = company_sales[company_sales.company.isin(test.index[0:5])]
				
				ex4)
				# データの準備
				attri_data2 = {'ID':['100','101','102','103','104'],
				'City':['Tokyo','Osaka','Kyoto','Hokkaido','Tokyo'],
				'Birth_year':[1990,1989,1992,1997,1982],
				'Name':['Hiroshi','Akiko','Yuki','Satoru','Steve']}
				attri_data_frame2 = DataFrame(attri_data2)
				attri_data_frame_index2 = DataFrame(attri_data2,index=['e','b','a','d','c'])
				
				# 値があるかどうかの確認
				attri_data_frame_index2.isin(['Tokyo'])
				#列ごとに、True, Falseが反る
				
				# 欠損値の取り扱い
				# name をすべてnanにする
				attri_data_frame_index2['Name'] = np.nan
				attri_data_frame_index2.isnull()
				
				# nullを判定し、合計する
				attri_data_frame_index2.isnull().sum()
				
				ex5)#特定の値を含むレコード数の確認
				# それぞれのカラムに ? が何個あるかカウント
				auto = auto[['price','horsepower','width','height']]
				auto.isin(['?']).sum()
					・DFに対して、リスト内にある値のみの行に絞る場合に使う
					Rのdplyr::filter(%in%)に近い
					
					Check whether values are contained in Series.
					
					Return a boolean Series showing whether each element in the Series matches an element in the passed sequence of values exactly.
						https://pandas.pydata.org/pandas-
						https://cmdlinetips.com/2018/02/hc	

39	pandas	method	info()	ex) nakamoto_corpus.info()	・pandas.DataFrame, pandas.Seriesの行数、列数、全要素数（サイズ）を取得する ・すべての変数について、nullでないデータの個数や変数の型がわかる	https://note.nkmk.me/python-panda	
40	base	method	help()	ex) help(sum)→OK help(info)→エラー help(nakamoto_test.info)→ヘルプが表示される	関数のヘルプを表示する。組み込み関数は関数名(括弧はつけない)を引数にすればOK。組み込み以外の関数は、インスタンス.関数名を引数にとる必要がある。		
41	base	method	len()	ex) l = [0, 1, 2, 3] print(len(l)) #4	リストの要素数確認ができる。	https://note.nkmk.me/python-list-le	
42	gensim	class?	corpora.Dictionary()	ex) dictionary = corpora.Dictionary(texts) print(dictionary) #Dictionary(15889 unique tokens: ['69', 'ペロリン', '反則', '飴ちゃん', '製麺']...)	gensimでLDAを行う際の辞書作成ができる。辞書には単語とIDのマッピングが含まれている。	http://sucrose.hatenablog.com/entr	

43	gensim.corpora.dictionary	method	doc2bow	ex)
				# コーパスを作成
				corpus = [dictionary.doc2bow(text) for text in texts] #list
				corpus_test = [dictionary.doc2bow(text) for text in texts_test]
					Convert `document` (a list of words) into the bag-of-words format = list of `(token_id, token_count)` 2-tuples. Each word is assumed to be a
					tokenized and normalized string (either unicode or utf8-encoded). No further preprocessing
					is done on the words in `document`; apply tokenization, stemming etc. before
					calling this method		

id	library/module	category	name	usage	remarks	url	備考

44	gensim.models.ldamodel	class?	gensim.models.
			ldamodel.
			LdaModel()
				ex)
				# LDA の計算
				topic_N = 1 + iterations
				parameters = topic_N
				lda = gensim.models.ldamodel.LdaModel(#gensim.models.ldamodel.LdaModel
				corpus=corpus,
				alpha='auto',
				num_topics=topic_N,
				id2word=dictionary)
					LDAのトピックモデル推定を行う		

45	gensim.models.ldamodel	method	lda.show_topics()	show_topics(num_topics=10, num_words=10, log=False, formatted=True) ex) lda.show_topics() #[(0, '0.011*"円" + 0.010*"麻婆豆腐" + 0.009*"方" + 0.008*"汗" + 0.008*"一" + 0.008*"前" + 0.007*"好き" + 0.007*"感じ" + 0.007*"席" + 0.006*"気"'), (1, '0.009*"メニュー" + 0.008*"1" + 0.008*"感じ" + 0.008*"蒙古タンメン中本" + 0.008*"----" + 0.007*"方" + 0.007*"O" + 0.007*"円" + 0.007*"店内" + 0.007*"目"')]	トピックごとの上位単語をリスト形式で返す The topics are returned as a list -- a list of strings if `formatted` is True, or a list of `(word, probability)` 2-tuples if False.	https://qiita.com/shizuma/items/44(	
46	pandas	インディケータ？	iloc()	ex) test = industry_market.iloc[:,1:]	DFから行、列指定でデータを切り出す		

47	pandas.core.reshape.concat	function	pd.concat()	concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None, levels=None, names=None, verify_integrity=False, sort=None, copy=True)
				
				ex)
				pd.concat([df_1, df_2], axis=1)
				
				#デフォルトでは行方向(bind_rows)に結合。sort=TRUEで結合されていない軸(今回は列)でのsortを行う
				dat = pd.concat([train,test],sort=True).reset_index(drop=True)
				
				ex2)複数DFを結合する場合
				※結合対象のDFは[]内で指定する必要がある
				dat = pd.concat([dat_1987, dat_1988, dat_1989])
					・DFを結合する
					・axis=1で横方向(bind_cols)。何も指定しないと(おそらく)縦方向
					・sort=TRUEで結合されていない軸名(今回は列名)でのsortを行う
					
					・DF結合の方法いろいろ
					https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html
						https://pandas.pydata.org/pandas-(	

48	-	記法	if文	if `条件式1`:
				`条件式1がTrueのときに行う処理`
				elif `条件式2`:
				`条件式1がFalseで条件式2がTrueのときに行う処理`
				elif `条件式3`:
				`条件式1, 2がFalseで条件式3がTrueのときに行う処理`
				...
				else:
				`すべての条件式がFalseのときに行う処理`
				
				ex)
				def if_test(num):
				if num > 100:
				print('100 < num')
				elif num > 50:
				print('50 < num <= 100')
				elif num > 0:
				print('0 < num <= 50')
				elif num == 0:
				print('num == 0')
				else:
				print('num < 0')
				
				# if文を1行で書きたい場合----------
				# ciperで暗号化した文字を復号化する関数
				def chiper_inv(text):
				res = ''
				for i in text:
				res += chr(219 - ord(i)) if i.islower() else i #if文を1行で書いた場合
				return res
						https://note.nkmk.me/python-if-elif-
						https://tutorial.djangogirls.org/ja/py	

id	library/module	category	name	usage	remarks	url	備考

49	numpy	method	arange()	ex)
				np.arange(start=1, stop=6, step=1)
				#array([1, 2, 3, 4, 5])
				
				#0～8までの整数
				np.arange(9)
				
				ex2)
				# pの値を0.001から0.999まで0.01刻みで動かす
				p = np.arange(0.001, 0.999, 0.01)
				
				# グラフ化
				plt.plot(p, calc_entropy(p))
				plt.xlabel('prob')
				plt.ylabel('entropy')
				plt.grid(True)
					・指定した連続した整数を発生する
					・スタートがnで差分がmの等差数列を作る。stopの番号は数列に含まれないので注意		

50	gensim.models.ldamodel	method	lda.show_topic()	show_topic(topicid, topn=10) ex) word_series = pd.DataFrame(lda.show_topic(i)).iloc[:, 0] #lda.show_topic(i)でi番目のトピックの上位10単語と割合を出す	トピックに属する上位n単語と確率を返す Return a list of `(word, probability)` 2-tuples for the most probable words in topic `topicid`. Only return 2-tuples for the topn most probable words (ignore the rest).		
51	progressbar	-	ProgressBar()	import progressbar import time min = 0 max = 100 prog_bar = progressbar.ProgressBar(min, max, widgets=[progressbar.CurrentTime(), ':', '(', progressbar.Counter(), ' of {}) '.format(100), progressbar.Bar(), ' ', progressbar.ETA(),]) for i in range(0, 100): prog_bar.update(i) time.sleep(1)	ループの進捗確認ができる	https://qiita.com/4hiziri/items/f1621	

52	-	記法	try-except文	ex)
				for topic_count in range(0, topic_number):
				try:
				result_parts.append(topic_distribution[document_count][topic_count][1])
				except IndexError: #稀にトピック割合が0のものがあり、タプルにデータそのものがないので「IndexError: list index out of range」になる
				result_parts.append(0) #トピックについての割合がない場合にはトピック割合に0を代入
					例外処理	https://www.sejuku.net/blog/23044	

53	nltk.stem.snowball	method	stem()	ex)
				#ステミング
				from nltk.stem.snowball import SnowballStemmer
				stemmer = SnowballStemmer("english")
				stemmer.stem("dogs are running")
				
				ex2)
				#文章を入れたら全部ステミングして返す関数
				def sentence_stemmer(sentence): #sentence例 "dogs are running"
				stemmer = SnowballStemmer("english")
				word_list = sentence.split()
				n = len(word_list)
				output = []
				
				for i in range(0, n):
				output.append(stemmer.stem(word_list[i]))
				
				return output
					単語を入れるとステム化した文章を返す
					
					stem(word) method of nltk.stem.snowball.EnglishStemmer instance
					Stem an English word and return the stemmed form.
					
					:param word: The word that is stemmed.
					:type word: str or unicode
					:return: The stemmed form.
					:rtype: unicode
						http://www.nltk.org/api/nltk.stem.ht	

id	library/module	category	name	usage	remarks	url	備考

54	-	記法	def文	ex)
				#文章を入れたら全部ステミングして返す関数
				def sentence_stemmer(sentence): #sentence例 "dogs are running"
				stemmer = SnowballStemmer("english")
				word_list = sentence.split()
				n = len(word_list)
				output = []
				
				for i in range(0, n):
				output.append(stemmer.stem(word_list[i]))
				
				return output
				
				# 掛け算をする関数
				def calc_multi(a, b):
				return a * b
				
				# **再帰関数**の例（フィボナッチ数）
				def calc_fib(n):
				if n == 1 or n == 2:
				return 1
				else:
				return calc_fib(n - 1) + calc_fib(n - 2)
					関数定義
					
					・次に示す関数は、フィボナッチ数を計算する例です。フィボナッチ数列とは、一歩手前と二歩手前の数を足して並べた数列のことをいいます（1, 1, 2, 3, 5...と前と前々の数字を足して、その数を並べたもの）。以下の関数calc_fibは、再帰と言って、自分の関数を中で呼び出しており、n番目のフィボナッチ数を作成しています。
						https://qiita.com/motoki1990/items/	

55	selenium	method	execute_script()	ex) driver.execute_script("window.scrollTo(0, document.body.scrollHeight)") #ページ最下部までスクロールする	ブラウザ上でjavscriptを実行する	https://stackoverflow.com/question	
56	base	関数	str()	ex) str(3040)	文字列変換	https://tutorial.djangogirls.org/ja/py	

57	base	メソッド	list.pop()	ex)
				>>> print(lottery)
				[59, 42, 30, 19, 12, 3, 199]
				>>> print(lottery[0])
				59
				>>> lottery.pop(0)
				59
				>>> print(lottery)
				[42, 30, 19, 12, 3, 199]
					リストの指定した値を削除する
					なお、辞書の値を削除するのにも同様のメソッドが使える
					
					Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)
						https://docs.python.org/3/tutorial/da	

58	base	演算子	and / or	ex)
				>>> 6 > 2 and 2 < 3
				True
				>>> 3 > 2 and 2 < 1
				False
				>>> 3 > 2 or 2 < 1
				True
					and – and の左辺と右辺が共にTrueの場合のみ、True。
					or – or の左辺あるいは右辺の少なくとも1つがTrueの時、True。
						https://tutorial.djangogirls.org/ja/py	

59	base	関数	range()	ex)
				for i in range(1, 6):
				print(i)
				#1~5の数字が表示される
				
				# range(N)とすると0からN-1までの整数
				for i in range(11):
				print(i)
				
				# range(1, 11, 2)は1から開始して2つ飛ばしで、11の手前まで取り出す
				for i in range(1, 11, 2):
				print(i)
					range 関数は、連続する数値を要素とするリストを作ります。引数に指定した開始の数値から終了の数値までのリストです。
					
					2つ目の引数（終了の数値）は、リストに含まれないことに注意してください。つまり、range(1, 6) は、1から5のことであり、6は含まれません。
						https://tutorial.djangogirls.org/ja/py	

60	仮想環境	仮想環境	virtualenv	ex)
				#仮想環境の作成
				$ python3 -m venv myvenv
				
				#仮想環境の起動
				$ source myvenv/bin/activate
					LinuxやOS Xでvirtualenvを作るときは、python3 -m venv myvenvと実行するだけ	https://tutorial.djangogirls.org/ja/dja	

id	library/module	category	name	usage	remarks	url	備考
61	Django	Django	manage.py runserver 0:8000	ex) (myvenv) ~/djangogirls$ python manage.py runserver 0:8000	webサーバの起動 Webサーバーが稼働している間は、追加のコマンドを入力するための新しいコマンドラインプロンプトは表示されません。新しいテキストを受け入れますが、新しいコマンドは実行しません。これは、Webサーバーが動作している間はずっとリクエストを待つためです。 Webサーバーの実行中に追加のコマンドを入力するには、新しいターミナルウィンドウを開き、virtualenvをアクティブにします。Webサーバーを停止するには、実行中のウィンドウに戻り、CTRL + C - ControlキーとCキーを同時に押します（WindowsではCtrl + Breakキーを押す必要があります）	https://tutorial.djangogirls.org/ja/dja	

62	(git)	(git)	(git)	ex)
				#gitリポジトリの初期化
				$ git init
				
				#状態の確認
				$ git status
				
				#このディレクトリ内の変更を全て反映させる
				$ git add --all .
				
				$ git commit -m "My Django Girls app, first commit"
				
				#自分のコンピューター上のGitリポジトリをGitHub上のGitリポジトリに結びつける
				$git remote add origin https://github.com/1kum1/my-first-blog.git
				#リモートレポジトリへ追加
				$ git push -u origin master
				
				#変更部分をpush
				$ git push
					gitリポジトリを初期化することは、プロジェクトごとに1回だけ行う必要があります（ユーザー名と電子メールをもう一度入力する必要はありません）
					
					git statusコマンドは、あらゆる追跡されていない/変更されている/ステージされている（untracked/modifed/staged）ファイルや、ブランチの状態などさまざまな情報を返します。
					
					コミットメッセージは"で囲む
					
					--all をつけると、git は、ファイルを削除したかどうかも判定します (これがない初期設定の状態では、新しいファイルと変更されたファイルしか認識しません)。. が、今いるディレクトリを表すということも思い出してくださいね(第3章にありました)。
						https://tutorial.djangogirls.org/ja/de
						https://tutorial.djangogirls.org/ja/htr	

63	(git)	(git)	.gitingnore		Git はこのディレクトリ内のすべてのファイルとフォルダの変更を追跡しますが、無視してほしいいくつかのファイルがあります。ベースディレクトリ内で .gitignore という名前のファイルを作成することによってこれを行います。 ・隠しファイルの一つ .gitignoreファイルに記載されたファイル及びディレクトリは変更してもgitに追跡されない	https://tutorial.djangogirls.org/ja/de	

64	pandas.core.frame.DataFrame	method	df.isnull()	ex)
				train.isnull().head()
				
				ex2)特定列の欠損値除外
				#CustomerIDにデータが入っているレコードのみに絞る
				excel_data_2 = excel_data[~excel_data.CustomerID.isnull()]
				
				#それぞれの行にNaNがいくつあるかの確認
				train.isnull().sum(axis = 1).head()
				#それぞれの列にNaNがいくつあるかの確認
				train.isnull().sum(axis = 0).head()
					・DFと同じサイズの論理値のDFを返す
					Return a boolean same-sized object indicating if the values are NA.
					NA values, such as None or :attr:`numpy.NaN`, gets mapped to True
					values.
					Everything else gets mapped to False values. Characters such as empty
					strings ``""`` or :attr:`numpy.inf` are not considered NA values
					(unless you set ``pandas.options.mode.use_inf_as_na = True``).		

65	pandas	method	Series.notnull()	ex) trans = trans[(trans.cancel_flg == '5') & (trans.CustomerID.notnull())]	・Detect existing (non-missing) values.		
66	組み込み	function	sum()	ex) train.isnull().sum()	・DFに適用するとseriesを返す		
67	pandas.core.frame	method	df.rename()	ex) #0を「欠損数」、1を「%」という名前に変更 kesson_table_ren_columns = kesson_table.rename(columns = {0 : '欠損数', 1 : '%'}) ex2) df_new = df.rename(columns={'A': 'a'}, index={'ONE': 'one'}) print(df_new) # a B C # one 11 12 13 # TWO 21 22 23 # THREE 31 32 33 print(df) # A B C # ONE 11 12 13 # TWO 21 22 23 # THREE 31 32 33	・DFの列名を変更する ・rename()メソッドの引数indexおよびcolumnsに、{元の値: 新しい値}のように辞書型で元の値と新しい値を指定する	https://note.nkmk.me/python-panda	

id	library/module	category	name	usage	remarks	url	備考

68	pandas.core.series	method	DF.fillna()	fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)
				
				ex)
				train["Age"] = train["Age"].fillna(train["Age"].median())
				
				ex2)
				#直前の値で埋める
				#金融系のデータなどで使えるらしい
				df.fillna(method = 'ffill')
				
				#平均値で穴埋めする場合
				df.fillna(df.mean())
					・seriesの欠損値をfillna()の括弧内で指定した値で埋める	https://deepage.net/features/pandas	

69	pandas.core.series	method	DF.ffill()	fx_jpusdata.resample('D').ffill().head()	・前のデータで穴埋めする	松尾研	

70	pandas.DataFrameの基本操作(列に対するもの)	その他	-	**#特定の1列を指定してseriesとして取り出す**
				#3列目を取り出す
				df_train_sur_age = df_train_sur.iloc[:,2] #列番号は0始まり
				#1～5行目を取り出す
				reserve_tb.iloc[:, 0:6]
				#列名指定で取り出す
				attri_data_frame_1.Birth_year
				
				# indexの値で指定してDFとして取り出す
				matrix[matrix.index == 2262708]
				
				#特定の1列を指定してarrayとして取り出す
				#機械学習モデルのX, yなどで使う
				y = student.G3.values
				
				#複数列を指定してndarrayとして取り出す
				student = pd.read_csv('student-mat.csv', sep=';')
				X = student.**loc**[:, ['age','Medu','Fedu','traveltime','studytime'
				,'failures','famrel','freetime','goout','Dalc','Walc'
				,'absences','G1','G2']].values
				
				#特定列に条件指定してデータを置き換える
				train["Sex"][train["Sex"] == "male"] = 0
				
				#特定列に条件を指定してデータを抽出 case1
				train[train["remarks"]!="お楽しみメニュー"]["y"]
				#こうするとグラフにもできる
				train[train["remarks"]!="お楽しみメニュー"]["y"].plot(figsize=(15,4))
				trainX = tmp[tmp["tt"]==1]
				attri_data_frame1[attri_data_frame1['City'] == 'Tokyo']
				#以下スクリプトはTrue, Falseを返す
				attri_data_frame1['City'] == 'Tokyo'
				#条件を複数指定する際
				attri_data_frame1[attri_data_frame1['City'].isin(['Tokyo','Osaka'])]
				
				#特定列に条件を指定してデータを抽出 case2
				#Survived列が1のデータを抽出
				df_train_sur = df_train_proc_dn[df_train_proc_dn.Survived==1]
				
				#複数列を指定してデータを切り出す
				features_one = train[["Pclass", "Sex", "Age", "Fare"]]
				
				#列名が特定の文字列であるものを除いたDFにする
				trainX.iloc[:, ~trainX.columns.str.match("y")]
				(dat.columns.str.match("y")はbool値のベクトルを返す)
				
				#列を追加してデータを挿入
				#name_countという列を追加してname_count_arrayの値を挿入
				df_train["name_count"] = name_count_array
				
				#特定の列(文字列型)にスライスを適用して文字列を切り出す
				print(df['a'].str[:2]) #先頭の文字を切り出す
					・df.query()もDFからのデータ抽出に使える	**https://cmdlinetips.com/2018/02/**
						http://pandas.pydata.org/pandas-d
						https://signate.jp/competitions/24/t
						http://www.mirandora.com/?p=180
						https://note.nkmk.me/python-panda	

id	library/module	category	name	usage	remarks	url	備考

71	DFの基本操作(行に対するもの)	その他	-	**#特定の1行を指定してseriesとして取り出す**
				df_train.iloc[2] #行番号は0始まり
				
				#行番号を指定してデータ抽出
				df_train.iloc[[0, 1, 2]] #行番号は0始まり #[]の中がarrayであること
				
				#行名を指定してデータを切り出す
				dat.index = pd.to_datetime(dat["datetime"]) #行名をつける
				dat = dat["2014-05-01":]
				dat = dat.reset_index(drop=True)
				
				#行番号を指定してデータを取り出す
				#0~2行目まで
				data.loc[0:2, :]
				
				#特定の行にデータを代入
				tr.loc[train_index,"tt"] = 1 #tt列を追加して、train_index(intのarray)で指定した行に1を代入
				tr.loc[test_index,"tt"] = 0 #tt列を追加して、test_indexで指定した行に0を代入
				
				#nullを含む行のみ抽出
				#df_trainのEmbarked列の値がnullの行のみ抽出
				df_train.Embarked[df_train.Embarked.isnull()]
				
				#nullを含む行のみ除外
				#trip_minutesがNAの行を除外
				dat_mod = dat[~dat.trip_minutes.isnull()]
				
				#条件を指定してデータ抽出
				trans = trans[(trans.cancel_flg == '5') & (trans.CustomerID.notnull())]
				
				#(時系列)特定の月を抽出
				#indexがtimestamp型の場合、以下で特定月のレコードが抽出可能
				time_series['2016-11']
				df = df['2015-01-01':'2015-12-31'] # 期間指定したい場合
					・df.query()もDFからのデータ抽出に使える	http://pandas.pydata.org/pandas-d
						https://signate.jp/competitions/24/t
						http://www.mirandora.com/?p=180	

72	pandas.Series	attribute	pandas.Series.values	ex) target = train["Survived"].values features_one = train[["Pclass", "Sex", "Age", "Fare"]].values	・pandas.Seriesの値をndarray(配列)などの型にして返す Return Series as ndarray or ndarray-like depending on the dtype. ・1次元であればベクトル、2次元であれば行列のような形式で返す	https://pandas.pydata.org/pandas-	

73	sklearn.tree.tree	class	DecisionTreeClassofoer	ex)
				from sklearn import tree
				
				# 「train」の目的変数と説明変数の値を取得
				target = train["Survived"].values #numpy.ndarray
				features_one = train[["Pclass", "Sex", "Age", "Fare"]].values #numpy.ndarray
				
				# 決定木の作成
				my_tree_one = tree.DecisionTreeClassifier()
				my_tree_one = my_tree_one.fit(features_one, target)
				
				# 「test」の説明変数の値を取得
				test_features = test[["Pclass", "Sex", "Age", "Fare"]].values
				
				# 「test」の説明変数を使って「my_tree_one」のモデルで予測
				my_prediction = my_tree_one.predict(test_features)
						https://scikit-learn.org/stable/modu
						https://www.codexa.net/kaggle-tita	

74	numpy	built-in function
		array in module numpy
			np.array()	array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)
				
				ex)
				PassengerId = np.array(test["PassengerId"]).astype(int) #numpy.ndarray
				
				#配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				# それぞれの数字を係数倍（ここでは2倍）
				data * 2
				
				# それぞれの要素同士での演算
				print('掛け算:', np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) * np.array([10, 9, 8, 7, 6, 5, 4, 3, 2, 1]))
				print('累乗:', np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) ** 2)
				print('割り算:', np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) / np.array([10, 9, 8, 7, 6, 5, 4, 3, 2, 1]))
				
				#3行3列の配列を作成
				matrix = np.array([[1,-1,-1], [-1,1,-1], [-1,-1,1]])
					creat an array
					
					・Pythonにおいて、Numpyではない、ふつうの配列（リスト）の、すべての要素を係数倍にするには、forを使ったループ処理が必要です。しかしNumpyの場合は、たとえば2倍にするのであれば、配列に対して「*2」と記述するだけで、すべての要素が2倍になりま
						松尾研	

id	library/module	category	name	usage	remarks	url	備考
75	numpy	method	array.astype()	array.astype(dtype, order='K', casting='unsafe', subok=True, copy=True) ex) PassengerId = np.array(test["PassengerId"]).astype(int) ex2) dat["precipitation"] = dat["precipitation"].apply(lambda x : -1 if x=="--" else x).astype(np.float)	・method of numpy.float64 instance ・method of numpy.ndarray instance ・Copy of the array, cast to a specified type		

76	pandas.core.tools.datetimes	function	pd.to_datetime()	to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, box=True, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=False) Convert argument to datetime.
				
				ex)
				#train.indexにdatetime列の値をdatetime型にして入れる
				train.index = pd.to_datetime(train["datetime"])
				train.head()
				
				ex)
				#特定列を日付型に変換
				reserve_tb["reserve_datetime"] = pd.to_datetime(reserve_tb["reserve_datetime"], format="%Y-%m-%d %H:%M:%S")
				#文字列を日付型に変換
				dat['month'] = pd.to_datetime(dat['month'], format='%Y-%m-%d')
				
				#年を取得
				※日付要素は列内にあるdtというオブジェクトが保持している
				reserve_tb["reserve_datetime"].dt.year
				
				#月を取得
				reserve_tb["reserve_datetime"].dt.month
				
				#日を取得
				reserve_tb["reserve_datetime"].dt.day
				
				#曜日(0:日曜日、1:月曜日)を数値で取得
				reserve_tb["reserve_datetime"].dt.dayofweek
				
				#時間を取得
				reserve_tb["reserve_datetime"].dt.hour
				reserve_tb["reserve_datetime"].dt.time
				
				#日付部分のみ取得
				excel_data2["InvoiceDate"].dt.date
					to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, box=True, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=False)
					Convert argument to datetime.
						前処理大全
						https://stackoverflow.com/question	

77	pandas.core.accessor	method	strftime()	# add month column dat['month'] = dat['date'].dt.**strftime**('%Y-%m-01') #月初の日付に直したい場合	・datetime型から文字列へ変換する	前処理大全 P239	

78	pandas.core.series	method	series.apply()	ex)
				# groupbyによるグルーピングとの併用
				f_max = lambda x: x.rolling(window=3, min_periods=1).max() #起点月含む過去3ヶ月の最大値
				train_monthly[{'item_cnt_%s' % 'max'}] = train_monthly.sort_values('date_block_num').groupby (['shop_id', 'item_category_id', 'item_id'])['item_cnt'].apply(f_max)
				※列追加するだけで、mappingできてないように見えるが、ちゃんとmappingできている
					・Invoke function on values of Series
					配列の値に関数を呼び出す
						https://www.kaggle.com/dimitreolive	

id	library/module	category	name	usage	remarks	url	備考

79	lambda式 (無名関数)	lambda式 (無名関数)	lambda式 (無名関数)	ex)
				def add_def(a, b=1):
				  return a + b
				add_lambda = lambda a, b=1: a + b
				
				train["precipitation"] = train["precipitation"].apply(lambda x : -1 if x == "--" else float(x))
				train["month"] = train["datetime"].apply(lambda x : int(x.split("-")[1]))
				
				ex2)
				sns.heatmap(cross_cluster_age.apply(lambda x : x/x.sum(), axis=1), cmap='Blues')
				
				ex3)
				# calc_multi関数を定義
				def calc_multi(a, b):
				  return a * b
				# それを実行 #30
				calc_multi(3, 10)
				#lambda式の場合
				(lambda a, b: a * b)(3, 10)
				#30
				
				list(map(lambda x : x * 2, [1, 2, 3, 4]))
					ここでlambda a, b:というのが、関数名(a, b)に相当する部分です。そして:で区切って、その関数の処理（ここではreturn a * b）を記述するというのが、無名関数の基本的な書き方です。
					無名関数は、リストなどの要素に対して何か関数を実行したいときに、よく使います。
						https://note.nkmk.me/python-lambd	

80	functools	function	reduce()	from functools import reduce
				
				def do_sum(x1, x2): return x1 + x2
				reduce(do_sum, [1, 2, 3, 4])
					・Apply a function of two arguments cumulatively to the items of a sequence, from left to right, so as to reduce the sequence to a single value.
					  For example, reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) calculates ((((1+2)+3)+4)+5).
					  If initial is present, it is placed before the items of the sequence in the calculation, and serves as a default when the sequence is empty.
					・関数を適用した結果にさらに関数を適用し、ベクトルからスカラーに減らす
						https://thepythonguru.com/python-l	

81	built-in	function	filter()	**Example 1: How filter() works for iterable list?**
				# list of alphabets
				alphabets = ['a', 'b', 'd', 'e', 'i', 'j', 'o']
				
				# function that filters vowels
				def filterVowels(alphabet):
				  vowels = ['a', 'e', 'i', 'o', 'u']
				
				  if(alphabet in vowels):
				  return True
				  else:
				  return False
				
				filteredVowels = filter(filterVowels, alphabets)
				
				print('The filtered vowels are:')
				for vowel in filteredVowels:
				  print(vowel)
				
				Example 2: How filter() method works without the filter function?
				# random list
				randomList = [1, 'a', 0, False, True, '0']
				
				filteredList = filter(None, randomList)
				
				print('The filtered elements are:')
				for element in filteredList:
				  print(element)
					・フィルタリングする関数を引数にとると、フィルタリングした結果を返す
					・フィルタリングする関数を引数にとらない場合は、True、Falseで絞った結果を返す
						https://www.programiz.com/python-	

id	library/module	category	name	usage	remarks	url	備考
82	matplotlib.pyplot	function	plt.subplots()	subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw) ex) fig, ax = plt.subplots(2,3,figsize=(9,6)) #figは白紙のグラフ、axは各グラフのindex train.plot.scatter(x="soldout", y="y", ax=ax[0][0]) train.plot.scatter(x="kcal", y="y", ax=ax[0][1]) train.plot.scatter(x="precipitation", y="y", ax=ax[0][2]) train.plot.scatter(x="payday", y="y", ax=ax[1][0]) train.plot.scatter(x="temperature", y="y", ax=ax[1][1]) train.plot.scatter(x="month", y="y", ax=ax[1][2]) plt.tight_layout()	・Create a figure with a set of subplots already made.		
83	built-in	function	dir()	ex) fig, ax = plt.subplots(2,3,figsize=(9,6)) dir(plt.subplots(2,3,figsize=(9,6)))	・変数のattribute等の一覧を返す If called without an argument, return the names in the current scope. Else, return an alphabetized list of names comprising (some of) the attributes of the given object, and of attributes reachable from it.		
84	一括コメントアウト(Jupyter)	-	-	macなら command + /			
85	pandas.plotting._core	method	df.plot.scatter()	scatter(x, y, s=None, c=None, **kwds) ex) fig, ax = plt.subplots(2,3,figsize=(9,6)) train.plot.scatter(x="soldout", y="y", ax=ax[0][0]) train.plot.scatter(x="kcal", y="y", ax=ax[0][1]) train.plot.scatter(x="precipitation", y="y", ax=ax[0][2]) train.plot.scatter(x="payday", y="y", ax=ax[1][0]) train.plot.scatter(x="temperature", y="y", ax=ax[1][1]) train.plot.scatter(x="month", y="y", ax=ax[1][2]) plt.tight_layout() ex2) import matplotlib.pyplot as plt import seaborn as sns plt.style.use('ggplot') df_train_sur = df_train_proc_dn[df_train_proc_dn.Survived==1] df_train_sur_age = df_train_sur.iloc[:,2] df_train_sur_s = df_train_sur.iloc[:,6] plt.scatter(df_train_sur_age,df_train_sur_s,color="#cc6699",alpha=0.5)	・Create a scatter plot with varying marker point size and color 散布図を描く	http://www.mirandora.com/?p=180	

86	seaborn.categorical	function	sns.boxplot()	boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, notch=False, ax=None, **kwargs)
				
				ex)
				fig, ax = plt.subplots(2,2,figsize=(12,7))
				sns.boxplot(x="week",y="y",data=train,ax=ax[0][0])
				sns.boxplot(x="weather",y="y",data=train,ax=ax[0][1])
				sns.boxplot(x="remarks",y="y",data=train,ax=ax[1][0])
				ax[1][0].set_xticklabels(ax[1][0].get_xticklabels(),rotation=30)
				sns.boxplot(x="event",y="y",data=train,ax=ax[1][1])
				plt.tight_layout()
				
				ex2)
				#3つの箱ひげ図を同時に描く場合
				# 箱ひげ図：G1,G2,G3
				plt.boxplot([student_data_math['G1'], student_data_math['G2'], student_data_math['G3']])
				plt.grid(True)
					・Draw a box plot to show distributions with respect to categories.
					箱ひげ図を描く
					・データに外れ値がある場合、それが省かれて、箱ひげ図が表示されるので注意
					・外れ値はデフォルトで指定されており、それを取り除いた場合のグラフが表示される		

87	matplotlib.axes._base	method	ax.set_xticklabels()	set_xticklabels(labels, fontdict=None, minor=False, **kwargs)
				
				ex)
				fig, ax = plt.subplots(2,2,figsize=(12,7))
				sns.boxplot(x="week",y="y",data=train,ax=ax[0][0])
				sns.boxplot(x="weather",y="y",data=train,ax=ax[0][1])
				sns.boxplot(x="remarks",y="y",data=train,ax=ax[1][0])
				ax[1][0].set_xticklabels(ax[1][0].get_xticklabels(),rotation=30) #x軸ラベルを斜めに表示
				sns.boxplot(x="event",y="y",data=train,ax=ax[1][1])
				plt.tight_layout()
					・Set the xtick labels with list of strings *labels*. Return a
					list of axis text instances.		

id	library/module	category	name	usage	remarks	url	備考

88	matplotlib.axes._base	method	ax.get_xticklabels()	get_xticklabels(minor=False, which=None)
				
				ex)
				fig, ax = plt.subplots(2,2,figsize=(12,7))
				sns.boxplot(x="week",y="y",data=train,ax=ax[0][0])
				sns.boxplot(x="weather",y="y",data=train,ax=ax[0][1])
				sns.boxplot(x="remarks",y="y",data=train,ax=ax[1][0])
				ax[1][0].set_xticklabels(ax[1][0].get_xticklabels(),rotation=30) #x軸ラベルを斜めに表示
				sns.boxplot(x="event",y="y",data=train,ax=ax[1][1])
				plt.tight_layout()
					・Get the x tick labels as a list of :class:`~matplotlib.text.Text` instances.		

89	scipy.stats.morestats	function	median_test()	median_test(*args, **kwds) ex) from scipy.stats import median_test stat,p,med,tbl = median_test(train[train["fun"]==1]["y"],train[train["fun"]==0]["y"]) print("p",p,"stat",stat)	・Mood's median test	https://docs.scipy.org/doc/scipy/ref	
90	built-in	method	find()	ex) #nameにカレーが入っていたら1、入ってなければ0を返す train["curry"] = train["name"].apply(lambda x : 1 if x.find("カレー")>=0 else 0) sns.boxplot(x="curry",y="y",data=train)	・method of builtins.str instance ・S.find(sub[, start[, end]]) -> int ・Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.		
91	markdownでインデントをつける	その他	その他	ex) * 方針としては日数が経過するにつれて減衰している為、売上数と日数の単回帰モデルを軸に検討する * 但し、2014-05以前はやや傾向が異なる為、学習データから除く * 他、お楽しみメニューやカレー等は大きく寄与はしていそうだが、非線形な関係であることも考慮し、Random Forestを用いて単回帰モデルの結果を修正するモデルも作成し、予測結果を導出することにする	・入れ子はインデントで表現する。タブ1つまたは半角スペース4つで認識する	https://qiita.com/higuma/items/334	

92	built-in	function	リストobj.index()	>>> l = [1, 3, 5, 2, 5]
				
				# 最大値のインデックス
				>>> l.index(max(l))
				2
					・リストの最大値、最小値を取得したい場合	https://hibiki-press.tech/learn_prog
						https://stackoverflow.com/question	

93	sklearnのバージョン確認	その他	.__version__	import sklearn print(sklearn.__version__) # 0.17.1		https://algorithm.joho.info/machine	
94	condaとpip：混ぜるな危険	その他	-	#自分の環境のパッケージ一覧を表示 !conda list #パッケージを探す !conda search scikit-learn		http://onoz000.hatenablog.com/ent	
95	sklearn.linear_model.base	class	LinerRegression() LR()	ex) >>> import numpy as np | >>> from sklearn.linear_model import LinearRegression | >>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]]) | >>> # y = 1 * x_0 + 2 * x_1 + 3 | >>> y = np.dot(X, np.array([1, 2])) + 3 | >>> reg = LinearRegression().fit(X, y) | >>> reg.score(X, y) | 1.0	・LinerRegressionクラスを作成する関数 ・LR()、LinerRegression()はおそらく同義		

id	library/module	category	name	usage	remarks	url	備考

96	線形回帰分析をしたい	線形回帰分析をしたい	線形回帰分析をしたい	from sklearn import linear_model
				
				# 線形回帰のインスタンスを生成
				reg = linear_model.LinearRegression()
				
				# 説明変数に "一期目の数学の成績" を利用
				# locはデータフレームから、行と列を指定して取り出す。loc[:, ['G1']]は、G1列のすべての列を取り出すことをしている
				# valuesに直しているので、注意
				X = student_data_math.loc[:, ['G1']].values
				
				# 目的変数に "最終の数学の成績" を利用
				Y = student_data_math['G3'].values
				
				# 予測モデルを計算、ここでa,bを算出
				reg.fit(X, Y)
				
				# 回帰係数
				print('回帰係数:', reg.coef_)
				
				# 切片
				print('切片:', reg.intercept_)
				
				# 先ほどと同じ散布図
				plt.scatter(X, Y)
				plt.xlabel('G1 grade')
				plt.ylabel('G3 grade')
				
				# その上に線形回帰直線を引く
				plt.plot(X, reg.predict(X))
				plt.grid(True)
				
				# 決定係数、寄与率とも呼ばれる
				print('決定係数:', reg.score(X, Y))
					・YY 、つまり予測したい最終の数学の成績G3は、predictを使って、括弧の中に説明変数を入れることで計算できます	松尾研	

97	重回帰分析をしたい	重回帰分析をしたい	重回帰分析をしたい	# データ分割（訓練データとテストデータ）のためのインポート
				from sklearn.model_selection import train_test_split
				
				# 重回帰のモデル構築のためのインポート
				from sklearn.linear_model import LinearRegression
				
				# 目的変数にpriceを指定、説明変数にそれ以外を指定
				X = auto.drop('price', axis=1)
				y = auto['price']
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				# 重回帰クラスの初期化と学習
				model = LinearRegression()
				model.fit(X_train,y_train)
				
				# 決定係数を表示
				print('決定係数(train):{:.3f}'.format(model.score(X_train,y_train)))
				print('決定係数(test):{:.3f}'.format(model.score(X_test,y_test)))
				
				# 回帰係数と切片を表示
				print('\n回帰係数\n{}'.format(pd.Series(model.coef_, index=X.columns)))
				print('切片: {:.3f}'.format(model.intercept_))
					・訓練時スコアとテスト時のスコアが近い場合、モデルは過学習に陥ってはいないと思われます
					
					■モデル構築とモデル評価の流れまとめ
					※決定木やSVMなどでも同じ
					・各種モデル構築のためのクラスのインスタンス化：model = LinearRegression()
					・データを説明変数と目的変数に分ける： XX と yy
					・訓練データとテストデータに分ける：train_test_split(X, y, test_size=0.5, random_state=0)
					・訓練データによるあてはめ（学習）：model.fit(X_train, y_train)
					・モデルの汎化性能をテストデータで確かめる：model.score(X_test, y_test)
						松尾研	

98	回帰モデルの評価をしたい	回帰モデルの評価をしたい	回帰モデルの評価をしたい		・URL参照 ・sklearn.metrics.SCORERS.keys() これで使える指標がわかる	https://scikit-learn.org/stable/module	

id	library/module	category	name	usage	remarks	url	備考

99	ロジスティック回帰をしたい	ロジスティック回帰をしたい	ロジスティック回帰をしたい	from sklearn.linear_model import LogisticRegression
				from sklearn.model_selection import train_test_split
				from sklearn.preprocessing import StandardScaler
				
				# 説明変数と目的変数の設定
				X = adult[['age','fnlwgt','education-num','capital-gain','capital-loss']]
				y = adult['fin_flg']
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				# 標準化処理
				sc = StandardScaler()
				sc.fit(X_train)
				X_train_std = sc.transform(X_train)
				X_test_std = sc.transform(X_test)
				
				# ロジスティック回帰クラスの初期化と学習
				model = LogisticRegression()
				model.fit(X_train_std,y_train)
				
				print('正解率(train):{:.3f}'.format(model.score(X_train_std, y_train)))
				print('正解率(test):{:.3f}'.format(model.score(X_test_std, y_test)))
				
				#学習済みモデルの各変数(age、fnlwgt、education-num、capital-gain、capital-loss**)の係数はcoef_属性を取得すると確認できる**
				model.coef_
				
				#それぞれのオッズ比は以下のように算出できます。
				#オッズ比とは、それぞれの係数が1増加したとき、正解率にどの程度影響があるかを示す指標
				np.exp(model.coef_)
				
				# テスト用データの予測確率を計算
				results = pd.DataFrame(model.predict_proba(X_test_std), columns=cancer.target_names)
					・高い性能が発揮されるのは線形分離可能なクラスに対してのみ
					・スケーリングによる予測精度の向上
					例えば、このモデルではage、fnlwgt、education-num、capital-gain、capital-losの5つの説明変数を使っていますが、それぞれの単位や大きさは異なっている
					このままだとモデルの学習が値の大きな変数に引っ張られ値の小さな変数の影響が小さくなる懸念がある
					そうならないようにするため、説明変数の標準化を実施し、説明変数の尺度を揃えることで、機械学習のアルゴリズムをよりうまく動作させられる
					
					・スケーリングには訓練データの平均値と標準偏差を使う
					テストデータの平均値、標準偏差を使ってはいけない
					テスト用データは将来手に入るであろう未知データという位置づけであるため
					
					・Scikit-learnのロジスティック回帰はデフォルトでq=2の正則化項が損失関数に含まれている(リッジ回帰)
					
					・ロジスティック回帰のパラメータチューニングの例
					https://www.kaggle.com/joparga3/2-tuning-parameters-for-logistic-regression
					Cの値を大きくし、正則化を弱めると、より複雑なモデルになり訓練データによくfitするようになる
						松尾研
						https://scikit-learn.org/stable/modu	

100	リッジ回帰をしたい	リッジ回帰をしたい	リッジ回帰をしたい	**※次のプログラムは、LinearRegressionクラスを使った重回帰モデル（linear）とRidgeクラスを使ったリッジ回帰モデル（ridge）を作り、その結果を比較するもの**
				
				# リッジ回帰用のクラス
				from sklearn.linear_model import Ridge
				from sklearn.model_selection import train_test_split
				
				# 訓練データとテストデータに分割
				X = auto.drop('price', axis=1)
				y = auto['price']
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				# モデルの構築と評価
				linear = LinearRegression()
				ridge = Ridge(random_state=0)
				
				for model in [linear, ridge]:
				  model.fit(X_train,y_train)
				  print('{}(train):{:.6f}'.format(model.__class__.__name__ , model.score(X_train,y_train)))
				  print('{}(test):{:.6f}'.format(model.__class__.__name__ , model.score(X_test,y_test)))
					・リッジ回帰についてはterm memory参照	松尾研	

id	library/module	category	name	usage	remarks	url	備考

101	ラッソ回帰をしたい	ラッソ回帰をしたい	ラッソ回帰をしたい	# ラッソ回帰と重回帰の比較
				# ラッソ回帰についてはパラメータを2種類試している
				from sklearn.linear_model import LinearRegression, Lasso
				
				X = auto.drop('price', axis=1)
				y = auto['price']
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				models = {
				'linear': LinearRegression(),
				'lasso1': Lasso(alpha=1.0, random_state=0),
				'lasso2': Lasso(alpha=200.0, random_state=0)
				}
				
				scores = {}
				for model_name, model in models.items():
				model.fit(X_train,y_train)
				scores[(model_name, 'train')] = model.score(X_train, y_train)
				scores[(model_name, 'test')] = model.score(X_test, y_test)
				
				pd.Series(scores).unstack()
					・ラッソ回帰についてはterm memory参照	https://scikit-learn.org/stable/module	

102	決定木のモデル構築がしたい	決定木のモデル構築がしたい	決定木のモデル構築がしたい	ex1)----------
				※DescriptionTreeClasifierクラスを使う際、パラメータのcriteroinに'entropy'を指定することで、分岐条件の指標としてエントロピーを設定しています
				
				from sklearn.tree import DecisionTreeClassifier
				from sklearn.model_selection import train_test_split
				
				# データ分割
				X = mushroom_dummy.drop('flg', axis=1)
				y = mushroom_dummy['flg']
				X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
				
				# 決定木クラスの初期化と学習
				model = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=0)
				model.fit(X_train,y_train)
				
				print('正解率(train):{:.3f}'.format(model.score(X_train, y_train)))
				print('正解率(test):{:.3f}'.format(model.score(X_test, y_test)))
				
				ex2)----------
				# パラメタ変更、不純度の指標変更をいろいろ試す場合
				from sklearn.tree import DecisionTreeClassifier
				
				cancer = load_breast_cancer()
				X_train, X_test, y_train, y_test = train_test_split(
				cancer.data, cancer.target, stratify = cancer.target, random_state=66)
				
				models = {
				'tree1': DecisionTreeClassifier(**criterion='entropy', max_depth=3,random_state=0**),
				'tree2': DecisionTreeClassifier(**criterion='entropy', max_depth=5, random_state=0**),
				'tree3': DecisionTreeClassifier(**criterion='entropy', max_depth=10, random_state=0**),
				'tree4': DecisionTreeClassifier(**criterion='gini', max_depth=3, random_state=0**),
				'tree5': DecisionTreeClassifier(**criterion='gini', max_depth=5, random_state=0**),
				'tree6': DecisionTreeClassifier(**criterion='gini', max_depth=10, random_state=0**)
				}
				
				scores = {}
				for model_name, model in models.items():
				model.fit(X_train,y_train)
				scores[(model_name, 'train')] = model.score(X_train, y_train)
				scores[(model_name, 'test')] = model.score(X_test, y_test)
				
				pd.Series(scores).unstack()
					・決定木についてはterm memory参照
					・sklearn.treeモジュールのDecisionTreeClassifierクラスを使うことで、決定木モデルを構築できます
					・決定木の分岐数決定のパラメータにmax_depthがあり、上記では5にしています。深ければ当然、条件分岐数の上限も増えます
					・正解率を高めるべくより複雑なモデルにしたい場合は深い木を作ればよいでしょう
					（ただし、あまり深い木を作ると過学習の危険性が増すので注意しましょう）
					・決定木は、モデルを構築する際に他のモデルでは必須となる標準化処理をしなくても結果は変わりません。
						松尾研
						https://scikit-learn.org/stable/module	

id	library/module	category	name	usage	remarks	url	備考

103	sklearn.metrics.classification	function	log_loss()	log_loss(y_true, y_pred, eps=1e-15, normalize=True, sample_weight=None, labels=None)
				
				ex)
				# For convenience we will use sklearn's GBM, the situation will be similar with XGBoost and others
				clf = GradientBoostingClassifier(n_estimators=5000, learning_rate=0.01, max_depth=3, random_state=0)
				clf.fit(X_train, y_train)
				
				y_pred = clf.predict_proba(X_test)[:, 1]
				print("Test logloss: {}".format(**log_loss**(y_test, y_pred)))
					This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of the true labels given a probabilistic classifier's predictions. The log loss is only defined for two or more labels. For a single sample with true label yt in {0,1} and estimated probability yp that yt = 1, the log loss is
					
					-log P(yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp))
						https://www.coursera.org/learn/com	

104	kNN(K近傍法)のモデル構築がしたい
	(分類タスク)
		kNN(K近傍法)のモデル構築がしたい
		(分類タスク)
			kNN(K近傍法)のモデル構築がしたい
			(分類タスク)
				# データやモデルを構築するためのライブラリ等のインポート
				from sklearn.datasets import load_breast_cancer
				from sklearn.neighbors import KNeighborsClassifier
				from sklearn.model_selection import train_test_split
				
				# データセットの読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				# stratifyは層化別抽出
				X_train, X_test, y_train, y_test = train_test_split(
				cancer.data, cancer.target, stratify = cancer.target, random_state=0)
				
				# グラフ描画用のリストを用意
				training_accuracy = []
				test_accuracy =[]
				
				# 学習
				# ここではkを1から20まで変化させ、訓練データとテストデータの正解率の変化を見ています
				for n_neighbors in range(1,21):
				# 分類タスクなので「KNeighborsClassifier」を使用
				# 回帰の場合は「KNeighborsRegressor」を使う
				model = **KNeighborsClassifier(**n_neighbors=n_neighbors)
				model.fit(X_train,y_train)
				training_accuracy.append(model.score(X_train, y_train))
				test_accuracy.append(model.score(X_test, y_test))
				
				# グラフを描画
				plt.plot(range(1,21), training_accuracy, label='Training')
				plt.plot(range(1,21), test_accuracy, label='Test')
				plt.ylabel('Accuracy')
				plt.xlabel('n_neighbors')
				plt.legend()
					・k-NNについてはterm memory参照
					・この例では、kが小さい時は正解率に乖離がありますが、6～8あたりで訓練とテストの正解率が近くなります。それ以上増やしてもモデル精度に大きな変化は見られません。精度に改善が見られない場合、あまりkを大きくする必要はないので、本ケースにおいては6～8程度に設定しておくのが良さそうです
						松尾研
						https://scikit-learn.org/stable/module	

105	kNN(K近傍法)のモデル構築がしたい
	(回帰)
		kNN(K近傍法)のモデル構築がしたい
		(回帰)
			kNN(K近傍法)のモデル構築がしたい
			(回帰)
				student = pd.read_csv('student-mat.csv', sep=';')
				X = student.loc[:,
				['age','Medu','Fedu','traveltime','studytime',
				'failures','famrel','freetime','goout','Dalc','Walc',
				'absences','G1','G2']].values
				
				# 解答
				from sklearn.neighbors import KNeighborsRegressor
				
				X_train, X_test, y_train, y_test = train_test_split(
				X, student.G3, random_state=0)
				
				scores_train = []
				scores_test =[]
				neighbors_settings = range(1, 20)
				for n_neighbors in neighbors_settings:
				model = KNeighborsRegressor(n_neighbors=n_neighbors)
				model.fit(X_train, y_train)
				scores_train.append(model.score(X_train, y_train))
				scores_test.append(model.score(X_test, y_test))
				
				plt.plot(neighbors_settings, training_accuracy,label='Training')
				plt.plot(neighbors_settings, test_accuracy,label='Test')
				plt.ylabel('R2 score') #回帰の場合はmodel.score(X_test, y_test)でaccuracyでなくR2 scoreが反る
				plt.xlabel('n_neighbors')
				plt.legend()
					・k-NNについてはterm memory参照	松尾研	

id	library/module	category	name	usage	remarks	url	備考

106	SVMのモデル構築がしたい
	(線形SVM)
		SVMのモデル構築がしたい
		(線形SVM)
			SVMのモデル構築がしたい
			(線形SVM)
				# SVMのライブラリ
				from sklearn.svm import **LinearSVC**
				# 訓練データとテストデータを分けるライブラリ
				from sklearn.model_selection import train_test_split
				# 標準化のライブラリ
				from sklearn.preprocessing import StandardScaler
				
				# データの読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				　cancer.data, cancer.target, stratify = cancer.target, random_state=0)
				
				# 標準化
				sc = StandardScaler()
				sc.fit(X_train)
				X_train_std = sc.transform(X_train)
				X_test_std = sc.transform(X_test)
				
				# クラスの初期化と学習
				model = **LinearSVC() #線形SVM**
				model.fit(X_train_std,y_train)
				
				# 訓練データとテストデータのスコア
				print('正解率(train):{:.3f}'.format(model.score(X_train_std, y_train)))
				print('正解率(test):{:.3f}'.format(model.score(X_test_std, y_test)))
					・サポートベクターマシンでは、標準化するとスコアが改善されることがあります	松尾研	

107	SVMのモデル構築がしたい
	(非線形SVM)
		SVMのモデル構築がしたい
		(非線形SVM)
			SVMのモデル構築がしたい
			(非線形SVM)
				# SVMのライブラリ
				from sklearn.svm import **SVC**
				
				# データの読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				　cancer.data, cancer.target, stratify = cancer.target, random_state=0)
				
				# 標準化
				sc = StandardScaler()
				sc.fit(X_train)
				X_train_std = sc.transform(X_train)
				X_test_std = sc.transform(X_test)
				
				# クラスの初期化と学習
				model = **SVC(kernel='rbf', random_state=0, C=2)**
				# gammma, Cというハイパーパラメタがあるのでそれを指定して実行してみても良い
				
				model.fit(X_train_std,y_train)
				
				# 訓練データとテストデータのスコア
				print('正解率(train):{:.3f}'.format(model.score(X_train_std, y_train)))
				print('正解率(test):{:.3f}'.format(model.score(X_test_std, y_test)))
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

108	複数モデル試したい	複数モデル試したい	複数モデル試したい	#必要なライブラリのインポート
				from sklearn.datasets import load_breast_cancer
				from sklearn.model_selection import train_test_split
				from sklearn.preprocessing import StandardScaler
				
				from sklearn.linear_model import LogisticRegression
				from sklearn.tree import DecisionTreeClassifier
				from sklearn.neighbors import KNeighborsClassifier
				from sklearn.svm import LinearSVC
				from sklearn.svm import SVC
				
				# データセットの読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				    cancer.data, cancer.target, stratify = cancer.target, random_state=0)
				
				# 標準化
				sc = StandardScaler()
				sc.fit(X_train)
				X_train_std = sc.transform(X_train)
				X_test_std = sc.transform(X_test)
				
				#学習するモデル
				models = {
				    'Logistic': LogisticRegression(),
				    'tree1': DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0),
				    'tree2': DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=0),
				    'tree3': DecisionTreeClassifier(criterion='entropy', max_depth=10, random_state=0),
				    'tree4': DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0),
				    'tree5': DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=0),
				    'tree6': DecisionTreeClassifier(criterion='gini', max_depth=10, random_state=0),
				    'LinerSVM': LinearSVC(),
				    'KernelSVM': SVC(kernel='rbf', random_state=0, C=2)
				}
				
				# こっちもあり
				# モデルクラスの初期化
				models = {
				** 'Logistic': LogisticRegression(random_state=0),**
				** 'SVM': SVC(kernel='linear', random_state=0),**
				** 'DecisionTree': DecisionTreeClassifier(random_state=0),**
				** 'KNN': KNeighborsClassifier(n_neighbors=5),**
				** 'RandomForest': RandomForestClassifier(random_state=0),**
				** 'GradientBoosting': GradientBoostingClassifier(random_state=0)**
				}
				
				scores = {}
				for model_name, model in models.items():
				    model.fit(X_train_std,y_train)
				    scores[(model_name, 'train')] = model.score(X_train_std, y_train)
				    scores[(model_name, 'test')] = model.score(X_test_std, y_test)
				
				pd.Series(scores).unstack()
					・dictionaryにモデルを入れてイテレータとして使える	松尾研	

id	library/module	category	name	usage	remarks	url	備考

109	kmeans法を用いたクラスタリングがしたい	kmeans法を用いたクラスタリングがしたい	kmeans法を用いたクラスタリングがしたい	ex1)
				# KMeansクラスの初期化
				kmeans = KMeans(init='random',n_clusters=3)
				
				# クラスターの重心を計算
				kmeans.fit(X)
				
				# クラスター番号を予測
				y_pred = kmeans.predict(X)
				
				# concatでデータを横に結合(axis=1を指定)
				merge_data = pd.concat([pd.DataFrame(X[:,0]), pd.DataFrame(X[:,1]), pd.DataFrame(y_pred)], axis=1)
				
				# 上記のデータにて、X軸をfeature1、Y軸をfeature2、クラスター番号をclusterと列名指定
				merge_data.columns = ['feature1','feature2','cluster']
				
				# クラスタリング結果のグラフ化
				ax = None
				colors = ['blue', 'red', 'green']
				for i, data in merge_data.groupby('cluster'):
				ax = data.plot.scatter(x='feature1', y='feature2', color=colors[i],
				label='cluster%d' % i, **ax=ax**)
				#ax = _x0008_axとすることで、1つのグラフ上にプロットされる
				#ax = ax_x0008_がない場合は、グラフがループの数だけ作られる
				
				ex2)----------
				#対象データを読み込み
				bank= pd.read_csv('bank-full.csv', sep=';')
				
				print('データ形式(X,y):{}'.format(bank.shape))
				print('欠損データの数:{}'.format(bank.isnull().sum().sum()))
				
				from sklearn.preprocessing import StandardScaler
				
				# データの列の絞り込み
				bank_sub = bank[['age','balance','campaign','previous']]
				
				# 標準化
				sc = StandardScaler()
				sc.fit(bank_sub)
				bank_sub_std = sc.transform(bank_sub)
				
				# KMeansクラスの初期化
				kmeans = KMeans(init='random', n_clusters=6, random_state=0)
				
				# クラスターの重心を計算
				kmeans.fit(bank_sub_std)
				
				# クラスター番号をpandasのSeriesオブジェクトに変換
				# クラスタリング処理を終えたら、kmeansオブジェクトのlabels_属性から、各データの所属クラスター番号を配列で取得できます
				labels = pd.Series(kmeans.labels_, name='cluster_number')
				
				# クラスター番号と件数を表示
				print(labels.value_counts(sort=False))
				
				# グラフを描画
				ax = labels.value_counts(sort=False).plot(kind='bar')
				ax.set_xlabel('cluster number')
				ax.set_ylabel('count')
					・クラスタ番号は0から始まる整数になる
					・まず、KMeansクラスを初期化しオブジェクトを作成します
					パラメータはinit='random'、n_clusters=3と設定しています
					initは初期化の方法。このようにrandomを設定するとk-means++ではなくk-means法となります。n_clustersにはクラスター数を設定します。
					
					・KMeansクラスのオブジェクトを作ったらfitメソッドを実行
					するとクラスターの重心が計算され、predictメソッドを実行することでクラスター番号が予測されます。
					・fitとpredictを一連の処理として実行するfit_predictメソッドもありますが、基本的に構築したモデルを保存する可能性のある場合は、fitメソッドを単独で実施するのがよい
					
					・変数の単位が異なる際は標準化をすると良い
					そうすることで値の大きな変数にクラスタリングの学習が引っ張られずに済みます。
					
					・クラスタ数の推定には**エルボー法**が使える
					下の行にサンプルコードがある
						松尾研	

110	エルボー法によるクラスタ数の推定がしたい	エルボー法によるクラスタ数の推定がしたい	エルボー法によるクラスタ数の推定がしたい	# エルボー方による推定。クラスター数を1から10に増やして、それぞれの距離の総和を求める
				dist_list =[]
				for i in range(1,10):
				kmeans= KMeans(n_clusters=i, init='random', random_state=0)
				kmeans.fit(X)
				dist_list.append(kmeans.inertia_)
				
				# グラフを表示
				plt.plot(range(1,10), dist_list,marker='+')
				plt.xlabel('Number of clusters')
				plt.ylabel('Distortion')
					・kmeansによるクラスタリングにおける、適切なクラスタ数の推定に用いられる
					・term memory参照
					・距離の総和はKMeansオブジェクトのinertia_属性で取得できます。クラスター数1から10までの距離の総和を求めてグラフにしたのが、次の図
					・エルボー法以外に、シルエット係数というものもある
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

111	主成分分析がしたい	主成分分析がしたい	主成分分析がしたい	**# 主成分分析の実行**
				# インポート
				from sklearn.decomposition import PCA
				
				# 主成分分析
				pca = PCA(n_components=2)
				pca.fit(X_std)
				
				# 学習結果の確認
				print(pca.components_)
				# [[-0.707 -0.707]
				# [-0.707 0.707]]
				
				print('各主成分の分散:{}'.format(pca.explained_variance_))
				
				print('各主成分の分散割合:{}'.format(pca.explained_variance_ratio_))
				
				# 結果の可視化
				# パラメータ設定
				arrowprops=dict(arrowstyle='->',
				linewidth=2,
				shrinkA=0, shrinkB=0)
				
				# 矢印を描くための関数
				def draw_vector(v0, v1):
				plt.gca().annotate('', v1, v0, arrowprops=arrowprops)
				
				# 元のデータをプロット
				plt.scatter(X_std[:, 0], X_std[:, 1], alpha=0.2)
				
				# 主成分分析の2軸を矢印で表示する
				for length, vector in zip(pca.explained_variance_, pca.components_):
				v = vector * 3 * np.sqrt(length)
				draw_vector(pca.mean_, pca.mean_ + v)
				
				plt.axis('equal');
				
				ex2)乳がんデータでの主成分分析
				# 乳がんデータを読み込むためのインポート
				from sklearn.datasets import load_breast_cancer
				
				# 乳がんデータの取得
				cancer = load_breast_cancer()
				
				# データをmalignant（悪性）かbenign（良性）に分けるためのフィルター処理
				# malignant（悪性）はcancer.targetが0
				malignant = cancer.data[cancer.target==0]
				
				# benign（良性）はcancer.targetが0
				benign = cancer.data[cancer.target==1]
				
				# malignant（悪性）がブルー、benign（良性）がオレンジのヒストグラム
				# 各図は、各々の説明変数（mean radiusなど）と目的変数との関係を示したヒストグラム
				fig, axes = plt.subplots(6,5,figsize=(20,20))
				ax = axes.ravel()
				for i in range(30):
				_,bins = np.histogram(cancer.data[:,i], bins=50)
				ax[i].hist(malignant[:,i], bins, alpha=.5)
				ax[i].hist(benign[:,i], bins, alpha=.5)
				ax[i].set_title(cancer.feature_names[i])
				ax[i].set_yticks(())
				
				# ラベルの設定
				ax[0].set_ylabel('Count')
				ax[0].legend(['malignant','benign'],loc='best')
				fig.tight_layout()
				
				# 標準化
				sc = StandardScaler()
				X_std = sc.fit_transform(cancer.data)
				
				# 主成分分析
				pca = PCA(n_components=2)
				pca.fit(X_std)
				X_pca = pca.transform(X_std)
				
				# 表示
				print('X_pca shape:{}'.format(X_pca.shape))
				print('Explained variance ratio:{}'.format(pca.explained_variance_ratio_))
				
				# 列にラベルをつける、1つ目が第1主成分、2つ目が第2主成分
					・主成分分析はsklearn.decompositionモジュールのPCAクラスを使うと実行できます。
					・オブジェクトの初期化の際、変数を何次元まで圧縮したいか、つまり、抽出したい主成分の数をn_componentsとして指定します。
					・通常は元ある変数よりも小さい値を設定します（30変数を5変数に減らす、等）が、ここでは元データと同じ2と設定します。
					・fitメソッドを実行することで、主成分の抽出に必要な情報が学習されます（具合的には、固有値と固有ベクトルが計算されます）。
					
					■components_属性
					components_属性は固有ベクトルと呼ばれるもので、主成分分析により発見された新しい特徴空間の軸の向きを表し、結果は以下になります。ベクトルの[-0.707,-0.707]が第1主成分、[-0.707,0.707]が第2主成分の向きになります。
					
					■explained_variance_属性
					explained_variance_属性は各主成分の分散を表します。以下を見ると、今回抽出された2つの主成分の分散が、それぞれ1.889と0.111であることがわかります。ここで分散の総和が2.0となるのは偶然ではなく、（標準化された）変数が元来有していた分散の総和と、主成分の分散の総和は一致します。つまり、分散（情報）は維持されているということです。
					
					■explained_variance_ratio_属性
					explained_variance_ratio_属性は、各主成分が持つ分散の比率です。最初の0.945は1.889/(1.889+0.111)によって得られ、第1主成分で元のデータの94.5%の情報を保持していると読めます。
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

112	アソシエーション分析(バスケット分析)がしたい	アソシエーション分析(バスケット分析)がしたい	アソシエーション分析(バスケット分析)がしたい	# すべてのInvoiceNoをtrans_allとして抽出
				trans_all = set(trans.InvoiceNo) #18536
				
				# 商品85123Aを購入したデータをtrans_aとする
				trans_a = set(trans[trans['StockCode']=='85123A'].InvoiceNo) #1978
				print(len(trans_a))
				
				# 商品85099Bを購入したデータをtrans_bとする
				trans_b = set(trans[trans['StockCode']=='85099B'].InvoiceNo)
				print(len(trans_b))
				
				# 商品85123Aおよび85099Bを購入したデータをtrans_abとする
				trans_ab = trans_a&trans_b
				print(len(trans_ab))
				
				# 全体のバスケットに占める商品Bの購買率を計算
				support_b = len(trans_b) / len(trans_all)
				
				# 商品Aを購買したときの商品Bの購買率を計算
				confidence = len(trans_ab) / len(trans_a)
				
				# リスト値を計算
				lift = confidence / support_b
				print('lift:{:.3f}'.format(lift))
					・アソシエーション分析についてはterm memory参照		

113	クロスバリデーション(cross validation, k分割交差検証法)したい	クロスバリデーション(cross validation, k分割交差検証法)したい	クロスバリデーション(cross validation, k分割交差検証法)したい	# 必要なライブラリ等のインポート
				from sklearn.datasets import load_breast_cancer
				from sklearn.tree import DecisionTreeClassifier
				from sklearn.model_selection import cross_val_score
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 決定木クラスの初期化
				tree = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
				
				# k分割交差検証の実行
				scores = cross_val_score(tree, cancer.data, cancer.target, cv=5) #cv=5は分割数
				
				# 結果の表示
				print('Cross validation scores: {}'.format(scores))
				print('Cross validation scores: {:.3f}+-{:.3f}'.format(scores.mean(), scores.std()))
					・基本的に、平均スコアの高いモデルを採用しますが、標準偏差が大きいときは、平均スコアから標準偏差を引いたスコアでモデルを選択してもよいでしょう。	松尾研 10章	

id	library/module	category	name	usage	remarks	url	備考

114	グリッドサーチ(grid search)がしたい	グリッドサーチ(grid search)がしたい	グリッドサーチ(grid search)がしたい	**ex1)SVMでのグリッドサーチ----------**
				from sklearn.model_selection import GridSearchCV
				from sklearn.svm import SVC
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(cancer.data,
				cancer.target,
				stratify = cancer.target,
				random_state=0)
				# GridSearchCVクラスに与えるパラメータを準備
				param_grid = { 'C': np.logspace(-3, 2, num=6)
				,'gamma':np.logspace(-3, 2, num=6)}
				
				# GridSearchCVクラスの初期化
				gs = GridSearchCV(estimator=SVC(),
				param_grid=param_grid,
				cv=5)
				
				# ハイパーパラメータの組み合わせの検証とベストモデルの構築
				gs.fit(X_train,y_train)
				
				# 表示
				print('Best cross validation score:{:.3f}'.format(gs.best_score_))
				print('Best parameters:{}'.format(gs.best_params_))
				print('Test score:{:.3f}'.format(gs.score(X_test,y_test)))
				
				ex2)決定木でのグリッドサーチ----------
				from sklearn.model_selection import GridSearchCV
				from sklearn.tree import DecisionTreeClassifier
				
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(cancer.data,
				cancer.target,
				stratify = cancer.target,
				random_state=0)
				
				# GridSearchCVクラスに与えるパラメータを準備
				param_grid = { 'max_depth': [2, 3, 4, 5]
				,'min_samples_leaf': [2, 3, 4, 5]}
				
				model = DecisionTreeClassifier(random_state=0)
				
				# GridSearchCVクラスの初期化
				gs = GridSearchCV(estimator= model,
				param_grid=param_grid,
				cv=5)
				
				# ハイパーパラメータの組み合わせの検証とベストモデルの構築
				gs.fit(X_train,y_train)
				
				# 表示
				print('Best cross validation score:{:.3f}'.format(gs.best_score_))
				print('Best parameters:{}'.format(gs.best_params_))
				print('Test score:{:.3f}'.format(gs.score(X_test,y_test)))
					・GridSearchCVクラスのfitメソッドの実行時に行われるモデルの評価は、デフォルトではk分割交差検証（厳密にはその改良版）が使われるという点です。そのためGridSearchCVクラスには初期化パラメータcvがあり、ここではcv=5と設定しました。	松尾研 10章	

id	library/module	category	name	usage	remarks	url	備考

115	混同行列(confusion matrix)を確認したい	混同行列(confusion matrix)を確認したい	混同行列(confusion matrix)を確認したい	# インポート
				from sklearn.svm import SVC
				from sklearn.metrics import confusion_matrix
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(cancer.data,
				cancer.target,
				stratify=cancer.target,
				random_state=66)
				# クラスの初期化と学習
				model = SVC(gamma=0.001,C=1)
				model.fit(X_train,y_train)
				
				# 表示
				print('{} train score: {:.3f}'.format(model.__class__.__name__, model.score(X_train,y_train)))
				print('{} test score: {:.3f}'.format(model.__class__.__name__ , model.score(X_test,y_test)))
				
				# ここからが混同行列作成の工程
				from sklearn.metrics import confusion_matrix
				
				# テストデータを使って予測値を算出
				y_pred = model.predict(X_test)
				
				m = confusion_matrix(y_test, y_pred)
				# 混同行列の出力
				print('Confution matrix:\n{}'.format(m))
				
				# 正解率
				accuracy = (m[0, 0] + m[1, 1]) / m.sum()
				print('正解率:{:.3f}'.format(accuracy))
				#正解率(別解)
				np.diag(confusion_matrix(y_test, y_pred)).sum()/confusion_matrix(y_test, y_pred).sum()
				
				# 適合率の計算
				precision = (m[1,1])/m[:, 1].sum()
				
				# 再現率の計算
				recall = (m[1,1])/m[1, :].sum()
				
				# F1スコアの計算
				f1 = 2 * (precision * recall)/(precision + recall)
				
				print('適合率:{:.3f}'.format(precision))
				print('再現率:{:.3f}'.format(recall))
				print('F1値:{:.3f}'.format(f1))
					・混同行列についてはterm memory参照
					・混合行列は、sklearn.metricsモジュールのconfusion_matrix関数で取得できます。出力される数値の並びは先の図で記した通り、列に予測値（y_pred）、行に観測値（y_test）が、負例・正例の順に並びます。
					・confusion_matrix()の結果は以下の形式で出力される
					列名部分が予測値の正例、負例
					行名部分が観測値の正例、負例
					|| 予測(0)|予測(1)|
					|------|------|------|
					| 観測(0) |48|5|
					| 観測(1) |8|82|
						松尾研 10章
						https://scikit-learn.org/stable/modu	

id	library/module	category	name	usage	remarks	url	備考

116	正解率 (accuracy)
	適合率 (precision)
	再現率 (recall)
	F1スコア
	を計算したい
		正解率
		(accuracy)
		適合率
		(precision)
		再現率 (recall)
		F1スコア
		を計算したい
			正解率 (accuracy)
			適合率 (precision)
			再現率 (recall)
			F1スコア
			を計算したい
				※「混同行列(confusion matrix)を確認したい」のusageのスクリプトの続き
				# インポート
				from sklearn.svm import SVC
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(cancer.data,
				cancer.target,
				stratify=cancer.target,
				random_state=66)
				# クラスの初期化と学習
				model = SVC(gamma=0.001,C=1)
				model.fit(X_train,y_train)
				
				# 表示
				print('{} train score: {:.3f}'.format(model.__class__.__name__, model.score(X_train,y_train)))
				print('{} test score: {:.3f}'.format(model.__class__.__name__ , model.score(X_test,y_test)))
				
				# テストデータを使って予測値を算出
				y_pred = model.predict(X_test)
				
				# ここからが各種評価指標の確認
				from sklearn.metrics import precision_score, recall_score, f1_score
				
				print('適合率:{:.3f}'.format(precision_score(y_test, y_pred)))
				print('再現率:{:.3f}'.format(recall_score(y_test, y_pred)))
				print('F1値:{:.3f}'.format(f1_score(y_test, y_pred)))
					正解率 (accuracy)
					適合率 (precision)
					再現率 (recall)
					F1スコア
					→term memory参照
					
					・注意)分類モデルの評価指標
						松尾研 10章	

117	ROC曲線を描きたい
	(2値分類)
		ROC曲線を描きたい
		(2値分類)
			ROC曲線を描きたい
			(2値分類)
				# インポート
				from sklearn import svm
				from sklearn.metrics import roc_curve, auc
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				cancer.data, cancer.target, test_size=0.5, random_state=66)
				
				# SVCによる予測確率の取得
				model = svm.SVC(kernel='linear', **probability=True**, random_state=0)
				#probability=Trueでないと、fitした際に予測確率(ROCを描く上で必要)が帰ってこない
				model.fit(X_train, y_train)
				
				# 予測確率を取得
				y_pred = model.predict_proba(X_test)[:,1]
				
				# 偽陽性率と真陽性率の算出
				fpr, tpr, thresholds = roc_curve(y_test, y_pred)
				
				# AUCの算出
				# aucは名前空間関連のエラーがよく起きるのでできるだけ厳密に指定する
				auc = sklearn.metrics.auc(fpr, tpr)
				
				# ROC曲線の描画
				plt.plot(fpr, tpr, color='red', label='ROC curve (area = %.3f)' % auc)
				plt.plot([0, 1], [0, 1], color='black', linestyle='--')
				
				plt.xlim([0.0, 1.0])
				plt.ylim([0.0, 1.05])
				plt.xlabel('False positive rate')
				plt.ylabel('True positive rate')
				plt.title('Receiver operating characteristic')
				plt.legend(loc="best")
					・注意)ROC, AUCは分類モデルの評価指標	松尾研 10章
						https://scikit-learn.org/stable/auto_	

id	library/module	category	name	usage	remarks	url	備考

118	ROC曲線を描きたい
	(多クラス分類)
		ROC曲線を描きたい
		(多クラス分類)
			ROC曲線を描きたい
			(多クラス分類)
				# 解答
				#参照URL：http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#sphx-glr-auto-examples-model-selection-plot-roc-py
				
				from sklearn import svm, datasets
				from sklearn.metrics import roc_curve, auc
				from sklearn.model_selection import train_test_split
				from sklearn.preprocessing import label_binarize
				from sklearn.multiclass import OneVsRestClassifier
				
				#　データの読み込み
				iris = datasets.load_iris()
				X = iris.data
				y = iris.target
				
				# 正解データのone-hot化
				y = label_binarize(y, classes=[0, 1, 2]) #01フラグ×3列に変形
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				# multi-class classification model
				model = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True, random_state=0))
				y_score = model.fit(X_train, y_train).predict_proba(X_test)
				
				# 3つそれぞれのクラスについて、1次元のデータにして、ROC曲線、AUCを算出する
				# 75サンプル×3列→225サンプル×1列の形式にすれば、2クラス分類同様にROC曲線が描ける
				fpr, tpr, _ = roc_curve(y_test.ravel(), y_score.ravel())
				roc_auc = sklearn.metrics.auc(fpr, tpr)
				
				# グラフ化する
				plt.figure()
				plt.plot(fpr, tpr, color='red', label='average ROC (area = {:.3f})'.format(roc_auc))
				plt.plot([0, 1], [0, 1], color='black', linestyle='--')
				plt.xlim([0.0, 1.0])
				plt.ylim([0.0, 1.05])
				plt.xlabel('False Positive Rate')
				plt.ylabel('True Positive Rate')
				plt.title('ROC')
				plt.legend(loc='best')
						松尾研 10章
						https://scikit-learn.org/stable/auto_	

119	バギング(bagging)を実装したい	バギング
		(bagging)を実装したい
			バギング(bagging)を実装したい	# インポート
				from sklearn.ensemble import BaggingClassifier
				# from sklearn.ensemble import BaggingRegressor #回帰タスクの場合はこちらを使う
				from sklearn.neighbors import KNeighborsClassifier
				from sklearn.model_selection import train_test_split
				
				# 乳がんのデータを読み込み
				cancer = load_breast_cancer()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				　cancer.data, cancer.target, stratify = cancer.target, random_state=66)
				
				# k-NNモデルとそのバギングの設定
				# n_estimatorsは、モデル何個でバギングするかという設定
				# 今回、100回k-NNモデルを作成
				models = {
				　'kNN': KNeighborsClassifier(),
				　'bagging': BaggingClassifier(KNeighborsClassifier(), n_estimators=100, random_state=0)
				}
				
				# モデル構築
				scores = {}
				for model_name, model in models.items():
				　model.fit(X_train, y_train)
				　scores[(model_name, 'train_score')] = model.score(X_train, y_train)
				　scores[(model_name, 'test_score')] = model.score(X_test, y_test)
				
				# 結果を表示
				pd.Series(scores).unstack()
					・バギングについてはterm memory
					・sklearn.ensembleモジュールのBaggingClassifierクラスを使っています。なお、回帰用のクラスもあるので、そちらについてはScikit-learnの公式ドキュメントを確認ください。
					・今回、k-NNモデル100個でバギングしている
					・BaggingClassifierクラスは他にmax_samples（デフォルトは1.0）、max_features（デフォルトは1.0）というパラメータを持ちます。
					　前者はブートストラップをする時に元のデータの何割抽出するかを指定します。0.5とすれば元の訓練データが100件あれば50件の標本が抽出されます。後者は説明変数をどの程度サンプリングするかの指定で、0.5とすれば全変数のうちの半分でモデルが学習されます。
					・元のモデルが過学習しているときは、手元のデータをそのまますべて使わないようにし、説明変数に（標本ごとに）多様性を与えるようにすることで、有効な過学習対策になる可能性があることを覚えておきましょう。
						松尾研 10章	

id	library/module	category	name	usage	remarks	url	備考

120	ブースティング(Boosting)を実装したい	ブースティング(Boosting)を実装したい	ブースティング(Boosting)を実装したい	# インポート
				from sklearn.tree import DecisionTreeRegressor
				from sklearn.ensemble import AdaBoostRegressor
				# from sklearn.ensemble import AdaBoostClassifier #分類タスクの場合はこちらを使う
				
				# housingデータを読み込み
				boston = load_boston()
				X_train, X_test, y_train, y_test = train_test_split(
				boston.data, boston.target, random_state=66)
				
				# 決定木とAdaBoostRegressorのパラメータ設定
				models = {
				'tree': DecisionTreeRegressor(random_state=0),
				'AdaBoost': AdaBoostRegressor(DecisionTreeRegressor(), random_state=0)
				}
				
				# モデル構築
				scores = {}
				for model_name, model in models.items():
				model.fit(X_train, y_train)
				scores[(model_name, 'train_score')] = model.score(X_train, y_train)
				scores[(model_name, 'test_score')] = model.score(X_test, y_test)
				
				# 結果を表示
				pd.Series(scores).unstack()
					・ブースティングについてはterm memory
					・AdaBoostRegressor()には回帰用のclassifierを入れないとエラーになる
					・AdaBoostClassofier()には分類用のclassifierを入れないとエラーになる
						松尾研 10章	

121	ランダムフォレスト
	勾配ブースティング
	を実装したい
		ランダムフォレスト
		勾配ブースティング
		を実装したい
			ランダムフォレスト
			勾配ブースティング
			を実装したい
				# インポート
				from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
				# 分類タスクの場合はRandomForestClassifier, GradientBoostingClassifier
				
				# Housingデータを読み込み
				boston = load_boston()
				
				# 訓練データとテストデータに分ける
				X_train, X_test, y_train, y_test = train_test_split(
				boston.data, boston.target, random_state=66)
				
				# ランダムフォレストと勾配ブースティングのパラメータ設定
				models = {
				'RandomForest': **RandomForestRegressor(random_state=0),**
				'GradientBoost': **GradientBoostingRegressor(random_state=0)**
				}
				
				# モデル構築
				scores = {}
				for model_name, model in models.items():
				model.fit(X_train, y_train)
				scores[(model_name, 'train_score')] = model.score(X_train, y_train)
				scores[(model_name, 'test_score')] = model.score(X_test, y_test)
				
				# 結果を表示
				pd.Series(scores).unstack()
				
				# ランダムフォレストのfeature importanceの確認
				# feature_importmnces属性を取得
				s = pd.Series(models['RandomForest'].feature_importances_,
				index=boston.feature_names)
				
				# 取得した値を降順に表示
				s.sort_values(ascending=False).plot.bar(color='C0')
					・変数の重要度は、(決定木構築の際の)情報利得をベースに計算されています。そのため変数の重要度を見ても回帰係数のような解釈は困難です。ただし、相対的な大小関係から、どの変数がモデル構築において重要であったかは示してくれますし、この変数が効果的であろうという直感と整合的な結果になることも少なくありません
					
					・Partial Dependence Plots（PDP）という、説明変数の大小と予測値の大小関係性を図解してくれる関数も存在します。Scikit-learnでは plot_partial_dependence関数が用意されています
						松尾研 10章
						https://scikit-learn.org/stable/modu
						
						https://scikit-learn.org/stable/modu
						
						https://scikit-learn.org/stable/modu	

id	library/module	category	name	usage	remarks	url	備考

122	XGBoostを実装したい	XGBoostを実装したい	XGBoostを実装したい	# モデル作成----------
				ts = time.time()
				
				model = XGBRegressor(
				max_depth=8,
				n_estimators=1000,
				min_child_weight=300,
				colsample_bytree=0.8,
				subsample=0.8,
				eta=0.3,
				seed=42)
				
				model.fit(
				X_train,
				Y_train,
				eval_metric="rmse",
				eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
				verbose=True,
				early_stopping_rounds = 10)
				
				time.time() - ts
				
				# 予測----------
				Y_pred = model.predict(X_valid).clip(0, 20)
				Y_test = model.predict(X_test).clip(0, 20)
				
				submission = pd.DataFrame({
				"ID": test.index,
				"item_cnt_month": Y_test
				})
				submission.to_csv('xgb_submission.csv', index=False)
				
				# save predictions for an ensemble
				pickle.dump(Y_pred, open('xgb_train.pickle', 'wb'))
				pickle.dump(Y_test, open('xgb_test.pickle', 'wb'))
				
				# Feature importance----------
				plot_features(model, (10,14))
						https://www.kaggle.com/dlarionov/fe	

123	LightGBMを実装したい	LightGBMを実装したい	LightGBMを実装したい	lgb_params = {
				'feature_fraction': 0.75,
				'metric': 'rmse',
				'nthread':1,
				'min_data_in_leaf': 2**7,
				'bagging_fraction': 0.75,
				'learning_rate': 0.03,
				'objective': 'mse',
				'bagging_seed': 2**7,
				'num_leaves': 2**7,
				'bagging_freq':1,
				'verbose':0
				}
				
				model = lgb.train(lgb_params, lgb.Dataset(X_train, label=y_train), 100)
				pred_lgb = model.predict(X_test)
				
				print('Test R-squared for LightGBM is %f' % r2_score(y_test, pred_lgb))
						Kaggle coursera	

124	sklearn.metrics	function	r2_score()	r2_score(y_true, y_pred, sample_weight=None, multioutput='uniform_average')	・R^2 (coefficient of determination) regression score function. ・決定係数の計算	https://scikit-learn.org/stable/module	

id	library/module	category	name	usage	remarks	url	備考
125	sklearn.ensemble.forest	class	RandomForestRegressor() RF()	ex) >>> from sklearn.ensemble import RandomForestRegressor | >>> from sklearn.datasets import make_regression | >>> | >>> X, y = make_regression(n_features=4, n_informative=2, | ... random_state=0, shuffle=False) | >>> regr = **RandomForestRegressor(max_depth=2, random_state=0, n_estimators=100)** | >>> regr.fit(X, y) | RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2, | max_features='auto', max_leaf_nodes=None, | min_impurity_decrease=0.0, min_impurity_split=None, | min_samples_leaf=1, min_samples_split=2, | min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, | oob_score=False, random_state=0, verbose=0, warm_start=False) | >>> print(regr.feature_importances_) | [0.18146984 0.81473937 0.00145312 0.00233767] | >>> print(regr.predict([[0, 0, 0, 0]])) | [-8.32987858]	・RandomForestRegressorクラスを作成する関数 (引数) ・n_estimator=10 数値。ランダムフォレスト内で使用する決定木の数 ・max_depth=None 木の深さの最大値。デフォルトなNone ・random_state=None "fit"と"predict"の両方に用いる並列化処理の数？		
126	sklearn.linear_model.base. LinearRegression	method	model.fit()	fit(X, y, sample_weight=None) ex) model1.fit(trainX["days"].values.reshape(-1,1),y_train)	・fit linear model 線形モデルをフィットさせる		
127	numpy.ndarray	method	array.reshape()	a.reshape(shape, order='C') ex) import numpy as np # 1つ1次元配列を生成。 a = np.arange(12) a #array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) # 3×4の2次元配列に変形。 b = np.reshape(a, (3, 4)) b #array([[0, 1, 2, 3], # [4, 5, 6, 7], # [8, 9, 10, 11]]) # 1行目の抽出 array1[0,:] # 1列目の抽出 array1[:,0]	・配列(array)の形状(何行何列)を変換する ・変換前の要素数と、変換後の要素数が一致しない場合はValueErrorの例外が発生します。 ・また、-1を使用すると、元の要素数に合わせて自動で適切な値が設定されます。 -1とした次元の長さは他の次元の指定値から推測されて自動的に決定される。サイズの大きい配列の形状を変換するときに便利。 ・行列から、行や列のみを抜き出したいときは、「[行範囲:列範囲]」のように表記	https://deepage.net/features/nump https://note.nkmk.me/python-nump	
128	sklearn.model_selection._split	class	KFold()	ex) kf = KFold(n_splits=5,random_state=777)	・KFoldクラスを作成する関数 (引数) ・n_splits=3 foldの数。少なくとも2以上は必要 ・random_state=None If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used ・shuffle : boolean, optional Whether to shuffle the data before splitting into batches.	https://blog.amedama.jp/entry/201	
129	sklearn.model_selection._split	method	KFold.split()	split(X, y=None, groups=None) ex)	・method of sklearn.model_selection._split.KFold instance ・Generate indices to split data into training and test set. indexを作成してデータを訓練用とテスト用に分ける		

id	library/module	category	name	usage	remarks	url	備考

130	pandas.core.reshape.reshape	function	pd.get_dummies()	get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None)
				
				ex)
				>>> df = pd.DataFrame({'A': ['a', 'b', 'a'], 'B': ['b', 'a', 'c'],
				... 'C': [1, 2, 3]})
				
				>>> pd.get_dummies(df, prefix=['col1', 'col2'])
				C col1_a col1_b col2_a col2_b col2_c
				0 1 1 0 0 1 0
				1 2 0 1 1 0 0
				2 3 1 0 0 0 1
				
				ex2)
				mushroom_dummy = pd.get_dummies(mushroom[['gill_color','gill_attachment','odor','cap_color']])
				mushroom_dummy.head()
					・配列やDFを入力とし、ダミー変数化されたDFを返す
					・Convert categorical variable into dummy/indicator variables
					・注)ダミー変数化対象の列に欠損値(NaN)が含まれていた場合、どのカテゴリにも属さないとして全てのダミー変数用列に0が入る		

131	del文	その他	-	ex) del trainX["tt"]	・要素を削除する Rだとrm()	https://www.javadrive.jp/python/list	
132	sklearn.linear_model.base	method	model.predict()	predict(X) ex) pred = model1.predict(trainX["days"].values.reshape(-1,1))	・method of sklearn.linear_model.base.LinearRegression instance (引数) ・X (n_samples, n_features)形式の配列かスパースmatrix		
133	sklearn.metrics.regression	function	mean_squared_error() MSE()	mean_squared_error(y_true, y_pred, sample_weight=None, multioutput='uniform_average') ex) MSE(y_train,pred_train)	・Mean squared error regression loss 平均二乗誤差		

134	pandas.core.frame	method	df.drop()	drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
				
				ex)
				sex_dum = pd.get_dummies(df_train['Sex'])
				df_train_proc = pd.concat((df_train,sex_dum),axis=1) #axis=1 でbind_cols
				df_train_proc = df_train_proc.drop('Sex',axis=1)
				df_train_proc = df_train_proc.drop('female',axis=1)
				
				# 複数行、複数列をまとめて削除
				print(df.drop(index=['Bob', 'Dave', 'Frank'],
				columns=['state', 'point']))
					・Drop specified labels from rows or columns
					特定の行あるいは列を削除する
					・axis=0 が行、axis=1 が列の削除
					・列(行)を削除しても元のデータの列(行)が削除されたわけではないので、注意
					置き換えたい場合は改めて変数に代入する
						https://note.nkmk.me/python-panda	

135	pandas.core.frame	method	df.dropna()	dropna(axis=0, how='any', thresh=None, subset=None, inplace=False) ex) df_train_proc_dn = df_train_proc.dropna() ex2)#特定の値を含む行の除外 # '?'をNaNに置換して、NaNがある行を削除 auto = auto.replace('?', np.nan).dropna() print('自動車データの形式:{}'.format(auto.shape))	・Remove missing values 欠損値を除外する ・DFを入力とし、欠損値(NA)を除外したDFを返す	http://www.mirandora.com/?p=180	
136	pandas.core.frame	method	df.corr()	corr(method='pearson', min_periods=1) ex) df_train_proc_dn = df_train_proc.dropna() df_train_proc_dn.corr() ex2) #特定のカラムについて、グルーピングしてグループ間の相関を見たい場合 data["stock_group_id"] = data.groupby(["stock"]).cumcount() data[["stock", "close", "stock_group_id"]].head(15) data.pivot(index = "stock_group_id", columns = "stock", values = "close").corr()	・Compute pairwise correlation of columns, excluding NA/null values 変数間の相関係数を計算する ・DFを入力として、DFを返す	http://www.mirandora.com/?p=180	
137	matplotlib	module	plt.style		・package matplotlib.style in matplotlib matplotlibライブラリにあるmatplotlod.styleパッケージ ・おそらく、グラフのスタイルに関する設定ができるパッケージ？	http://www.mirandora.com/?p=180	

id	library/module	category	name	usage	remarks	url	備考
138	matplotlib.style.core	function	plt.style.use()	use(style) ex) import matplotlib.pyplot as plt import seaborn as sns plt.style.use('ggplot') df_train_sur = df_train_proc_dn[df_train_proc_dn.Survived==1] df_train_sur_age = df_train_sur.iloc[:,2] df_train_sur_s = df_train_sur.iloc[:,6] plt.scatter(df_train_sur_age,df_train_sur_s,color="#cc6699",alpha=0.5) df_train_sur = df_train_proc_dn[df_train_proc_dn.Survived==0] df_train_sur_age = df_train_sur.iloc[:,2] df_train_sur_s = df_train_sur.iloc[:,6] plt.scatter(df_train_sur_age,df_train_sur_s,color="#6699cc",alpha=0.5) plt.show()	・Use matplotlib style settings from a style specification.	http://www.mirandora.com/?p=180	

139	pandas.core.frame	method	df.append	append(other, ignore_index=False, verify_integrity=False, sort=None)
				
				ex)
				def name_classifier(name_df):
				　name_class_df = pd.DataFrame(columns={'miss','mrs','master','mr'})
				
				　for name in name_df:
				　　if 'Miss' in name:
				　　　df = pd.DataFrame([[1,0,0,0]],columns={'miss','mrs','master','mr'})
				　　elif 'Mrs' in name:
				　　　df = pd.DataFrame([[0,1,0,0]],columns={'miss','mrs','master','mr'})
				　　elif 'Master' in name:
				　　　df = pd.DataFrame([[0,0,1,0]],columns={'miss','mrs','master','mr'})
				　　elif 'Mr' in name:
				　　　df = pd.DataFrame([[0,0,0,1]],columns={'miss','mrs','master','mr'})
				　　else :
				　　　df = pd.DataFrame([[0,0,0,0]],columns={'miss','mrs','master','mr'})
				　　name_class_df = name_class_df.append(df,ignore_index=True)
				　return name_class_df
					・Append rows of `other` to the end of this frame, returning a new object. Columns not in this frame are added as new columns
					　対象のDF末尾に別のrowを追加し、新しいオブジェクトとして返す。
					　対象のDFに存在しないカラムは新たなカラムとして追加される。
					
					引数)
					・ignore_index = FALSE
					　bool値を指定する。TRUEであれば、indexラベルを使わない
						http://www.mirandora.com/?p=180	

140	built-in	method	strオブジェクト. replace()	replace(...) S.replace(old, new[, count]) -> str ex) 'Braund★'.replace('★','') #Braund	・method of builtins.str instance 　組み込みのstrインスタンスのメソッド ・Return a copy of S with all occurrences of substring old replaced by new.	http://www.mirandora.com/?p=180	
141	built-in	method	list.index()	index(...) L.index(value, [start, [stop]]) -> integer ex) bet = "best" test = ["better", "best", "good"] test.index(bet) # 1	・method of builtins.list instance ・return first index of value. 　()内で指定した値の最初のindex(数字)を返す ・Raises ValueError if the value is not present.	http://www.mirandora.com/?p=180	
142	pandas.core.frame	method	df.query()	query(expr, inplace=False, **kwargs) ex) perfect_df_other = perfect_df.query("master==0 and mr==0 and mrs==0 and miss==0")	・Query the columns of a frame with a boolean expression. 　論理表現でDFの列に問い合わせする	http://www.mirandora.com/?p=180	

143	printで計算結果の数値を代入した文章を表示する	その他	print()	ex)
				# データ表示（特徴量）
				print("データ数 = %d　特徴量 = %d" % (X.shape[0], X.shape[1]))
				pd.DataFrame(X, columns=boston.feature_names).head()
						https://qiita.com/fujin/items/128ed7	

id	library/module	category	name	usage	remarks	url	備考

144	pandas.core.generic	method	df.groupby()	groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, observed=False, **kwargs)
				
				ex)
				# データのグループ集計
				attri_data_frame2.groupby('Sex')['Math'].mean() #結果はseries
				attri_data_frame2.groupby('Sex')[['Math']].mean() #結果はdataframe
				
				ex2)
				#データ準備
				data1 = {
				'id': ['100', '101', '102', '103', '104', '106', '108', '110', '111',' 113'],
				'city': ['Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo', 'Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo'],
				'birth_year': [1990, 1989, 1992, 1997, 1982, 1991, 1988, 1990, 1995, 1981],
				'name': ['Hiroshi', 'Akiko', 'Yuki', 'Satoru', 'Steeve', 'Mituru', 'Aoi', 'Tarou', 'Suguru','Mitsuo']
				}
				df1 = DataFrame(data1)
				
				# 出現頻度
				df1.groupby('city').**size()**
				
				# cityを軸に、birth_yearの平均値を求める
				df1.groupby('city')['birth_year'].mean()
				
				#軸は複数設定することもできます
				#たとえば、region、cityを2軸として、birth_yearの平均値を求めると、次のようになります。
				df1.groupby(['region', 'city'])['birth_year'].mean()
				
				#as_index = Falseパラメータを設定すると、インデックスが設定されなくなります
				#そのままテーブルとして扱いたいときに便利です。
				# reset_index()をやったのとおなじになる
				df1.groupby(['region', 'city'], **as_index = False**)['birth_year'].mean()
				
				#groupbyメソッドには、イテレータという、反復的に値を取り出す機能があり、次のように、結果の要素をPythonのforなどでループ処理できて便利
				for group, subdf in df1.groupby('region'):
				print('==')
				print('Region Name:{0}'.format(group))
				print(subdf)
				
				ex3)#グループごとに連番を振る
				data["stock_group_id"] = data.groupby(["stock"]).cumcount()
				
				ex4)aggで集約関数をまとめて適用する
				sales.groupby(index_cols,as_index=False).agg({'item_cnt_day':'sum', 'hogefuga':'nunique'})
				monthly_sales=sales.groupby(["date_block_num","shop_id","item_id"])[
				"date","item_price","item_cnt_day"].agg({"date":["min","max"],"item_price":"mean","item_cnt_day":"sum"})
					・Group series using mapper (dict or key function, apply given function
					to group, return result as series) or by a series of columns.
						http://pandas.pydata.org/pandas-d
						https://stackoverflow.com/question	

145	pandas.core.groupby	method	df.groupby().cumsum()	cumsum(axis=0, *args, **kwargs) ・method of pandas.core.groupby.SeriesGroupBy instance ・Cumulative sum for each group ex) test = pd.DataFrame({'item_id': [1, 1, 1, 2, 2, 2, 2], 'target': [3, 4, 5, 6, 7, 8, 9]}) test test.groupby('item_id')['target'].cumsum()	・グループごとに累積和を出す		

146	pandas.core.groupby.groupby	method	df.groupby("test").size()
			
			df.groupby("test").count()
				ex)
				#Embarkedの最頻値を確認
				tab = df_train.groupby('Embarked').size()
				tab
					・Compute group sizes
					グールプのサイズを計算
					・count()でも同じ
					・size()はNaNを含み、count()はNaNを含まない点が違い
						https://stackoverflow.com/questions/	

147	pandas.core.groupby.groupby	method	df.groupby.shift()	ex) train_monthly['item_cnt_month'] = train_monthly.sort_values('date_block_num').groupby(['shop_id', 'item_id'])['item_cnt'].shift(-1)	・グループごとにデータをn行ずらす ・グルーピングしてデータをずらすだけで、集約はしてないので、元のDFに列として追加できる	https://stackoverflow.com/question https://www.kaggle.com/kernels/sc	

id	library/module	category	name	usage	remarks	url	備考

148	pandas.core.groupby	method	df.groupby().
			transform()
				# mean-encodingに便利
				all_data['item_target_enc'] = all_data.groupby('item_id')['target'].transform('mean')
				
				# 以下のスクリプトでも結果は同じ
				# item_id_target_mean = all_data.groupby('item_id').target.mean()
				# all_data['item_target_enc'] = all_data['item_id'].map(item_id_target_mean)
				
				all_data.groupby('item_id')['target'].mean() だと、単に集計するだけなので、マッピング処理が別途必要
				
				# leave one out schemeのtarget encodingに便利
				test = pd.DataFrame({'item_id': [1, 1, 1, 2, 2, 2, 2], 'target': [3, 4, 5, 6, 7, 8, 9]})
				# groupbyオブジェクト(Series)を渡すと、Looスキームで処理したtarget encoding結果を返す関数
				loo = lambda x: (x.sum() - x)/(len(x) - 1)
				test.groupby('item_id')['target'].transform(loo)
					・groupbyした結果に関数を適用し、マッピング処理をする	https://deepage.net/features/panda	

149	sklearn.model_selection. train_test_split	function	train_test_split()	train_test_split(*arrays, **options) ex)	・Split arrays or matrices into random train and test subsets ・テストサイズはデフォルトで0.25		
150	np.nan	np.nan	np.nan			https://takala.tokyo/takala_wp/201	
151	seaborn.axisgrid	function	sns.pairplot()	pairplot(data, hue=None, hue_order=None, palette=None, vars=None, x_vars=None, y_vars=None, kind='scatter', diag_kind='hist', markers=None, size=2.5, aspect=1, dropna=True, plot_kws=None, diag_kws=None, grid_kws=None) ex) from matplotlib import pyplot as plt import seaborn as sns %matplotlib inline sns.pairplot(dat_mod[["trip_minutes", "trip_miles", "fare"]], **size=3, aspect=2**, diag_kind="kde")	・散布図行列を作成する ・DFにnullを含むカラムがあるとエラーになることがある ■引数について ・size→高さ ・size*aspect→横幅 ・diag_kind→対角線上のグラフ種類の指定		

152	seaborn.distributions	function	sns.distplot()	distplot(a, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=None, kde_kws=None, rug_kws=None, fit_kws=None, color=None, vertical=False, norm_hist=False, axlabel=None, label=None, ax=None)
				
				ex)
				from matplotlib import pyplot as plt
				import seaborn as sns
				sns.set() #グラフのテーマを見やすいものに変更
				%matplotlib inline
				
				plt.xlim([0, 200]) #x軸範囲を指定する場合
				plt.ylim([0, 1200]) #y軸範囲を指定する場合
				sns.distplot(fish_data, **bins=200**, kde=False) #binsの値は大きいほど階級が狭くなる
				#fish_dataはseries
				
				ex)
				#複数のグラフを並べる場合
				fig = plt.figure(figsize=(10, 7))
				
				fig.add_subplot(221)
				plt.xlim(0, 100)
				sns.distplot(dat_mod["trip_minutes"], kde=False, bins=600) #1行1列目
				
				fig.add_subplot(222)
				plt.xlim(0, 30)
				sns.distplot(dat_mod["trip_miles"], kde=False, bins=800) #1行2列目
				
				fig.add_subplot(223)
				plt.xlim(0, 100)
				sns.distplot(dat_mod["fare"], kde=False, bins=2000) #2行1列目
					・ヒストグラムを描画する
					・bins 引数でビンの数を指定できる。
						http://pynote.hatenablog.com/entry
						https://python-analytics.hatenadiar	

153	seaborn.axisgrid	function	sns.jointplot()	jointplot(x, y, data=None, kind='scatter', stat_func=<function pearsonr at 0x1820998268>, color=None, size=6, ratio=5, space=0.2, dropna=True, **xlim=None, ylim=None**, joint_kws=None, marginal_kws=None, annot_kws=None, **kwargs) ex) sns.jointplot(x="trip_minutes", y="trip_miles", data=dat_mod[["trip_minutes", "trip_miles"]], **xlim=(0, 800), ylim=(0, 200)**)	・jointplotを描画する 2変量散布図と周辺分布を描いたもの		

id	library/module	category	name	usage	remarks	url	備考

154	組み込み	function	vars()	ex)
				class Test:
				  def __init__(self):
				  self.title = '題名'
				  self.name = '名前'
				
				test = Test()
				
				print(vars(test))
				
				{'name': '名前', 'title': '題名'}
						http://www.ajisaba.net/python/class	

155	pandas		DF.dtypes	ex) taxi_duration.dtypes # pickup_community_area int64 # record_count int64 # minutes_per_mile_avg float64	・DFのデータ型を確認	https://note.nkmk.me/python-panda	
156	pandas.core.generic	function	DF.astype()	astype(dtype, copy=True, errors='raise', **kwargs) ex) #yearを型変換 dat_mod["year"] = dat_mod["year"].astype(str) print(dat_mod.year.dtype) #int #pickup_community_areaをカテゴリ変換 taxi_duration["pickup_community_area"] = taxi_duration["pickup_community_area"].astype("category") #float型に変換 data["open"] = data["open"].astype(float)	Cast a pandas object to a specified dtype ``dtype`` ・pandas dataframeの型キャスト	https://note.nkmk.me/python-panda	

157	seaborn.relational	function	sns.scatterplot()	scatterplot(x=None, y=None, hue=None, style=None, size=None, data=None, palette=None,
				hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=True,
				style_order=None, x_bins=None, y_bins=None, units=None, estimator=None, ci=95, n_boot=1000,
				alpha='auto', x_jitter=None, y_jitter=None, legend='brief', ax=None, **kwargs)
				
				ex)
				ax = sns.scatterplot(x=dat_mod.trip_minutes, \
				  y=dat_mod.trip_miles, \
				  hue=dat_mod.start_timestamp.dt.year, \ #hueでグループ化に使う変数を指定
				  style=dat_mod.start_timestamp.dt.year) #styleでスタイルを分ける変数を指定
				
				ax.set_xlim(0, 200)
				ax.set_ylim(0, 30)
					・seaborn.__version__ >= 9.0 でないと使えない	https://qiita.com/skotaro/items/08dc	

158	pandas.core.reshape.pivot	function	pd.pivot_table()	pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None,
				margins=False, dropna=True, margins_name='All')
				
				ex)
				#タクシー会社ごとの売上げランキング
				test = pd.pivot_table(
				  data = company_sales,
				  values = "sales",
				  aggfunc = "sum",
				  index = "company")
				
				ex2)
				#プロット
				pd.pivot_table(data = plot_data, values = "close", columns = "stock", index = "date").plot(figsize = (10,
				7), grid = True)
					・Create a spreadsheet-style pivot table as a DataFrame.
					  The levels in the pivot table will be stored in MultiIndex objects
					(hierarchical indexes) on the index and columns of the result
					DataFrame
						help(pd.pivot_table)	

159	pandas.core.groupby.groupby	method	DF.groupby
			("column").
			aggragate()
				aggregate(arg, *args, **kwargs)
				
				ex)
				test2 = company_sales.groupby("company").agg({"sales":"sum"})
				
				ex2)
				# 列に複数の関数を適応
				functions = ['count','mean','max','min']
				grouped_student_math_data1 = student_data_math.groupby(['sex','address'])
				grouped_student_math_data1['age','G1'].agg(functions)
					・aggregate using one or more operations over the specified axis
					・aggメソッドの引数には、実行したい関数名のリストを渡します。例は、カウント、平均、最大、最小を計算する場合
						help(DF.groupby("column").agg)	

id	library/module	category	name	usage	remarks	url	備考

160	pandas.core.frame	method	DF.sort_values()	sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
				
				ex)
				test = company_sales.groupby("company").agg({"sales":"sum"}).sort_values("sales", ascending=False)
				
				ex2)
				# 値によるソート、デフォルトは昇順
				attri_data_frame_index2.Birth_year.sort_values()
				
				log_timeline.std().sort_values(ascending=False) #降順でソート
				
				ex3)
				excel_data2.groupby(["Country"], as_index=False)["sales"].agg("sum").sort_values(["sales"], ascending=False)
				
				ex4)#2軸で並び替えをする
				top_five_country_description_sales.sort_values(["Country", "sales"], ascending=[True, False])
					・Sort by the values along either axis
					・pandas seriesにも使える
					・行名でソートする場合はsort_index()
					
					■引数
					・inplace = True → 元の変数を上書きする
						https://pandas.pydata.org/pandas-do	

161	pandas.core.strings	method	DF.*column_name*.str.lower()	ex) company_sales["company"] = company_sales.company.str.lower()	・Convert strings in the Series/Index to lowercase 配列やindexの文字列を小文字変換する	https://stackoverflow.com/questions/	
162	pandas.core.strings	method	DF.*column_name*.str.upper()	ex) company_sales["company"] = company_sales.company.str.upper()	・Convert strings in the Series/Index to uppercase 配列やindexの文字列を大文字変換する	https://stackoverflow.com/questions/	
163	seaborn.relational	function	sns.lineplot()	lineplot(x=None, y=None, hue=None, size=None, style=None, data=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, dashes=True, markers=None, style_order=None, units=None, estimator='mean', ci=95, n_boot=1000, sort=True, err_style='band', err_kws=None, legend='brief', ax=None, **kwargs) ex) sns.set(style="whitegrid") fig = plt.figure(figsize=(10, 7)) ax = sns.lineplot(x="month", y="sales", hue="company", style="company", data=company_sales_top5, linewidth=2.5)	・Draw a line plot with possibility of several semantic groupings		
164	seaborn	function	sns.barplot()	ex) fig = plt.figure(figsize=(20, 10.5)) sns.barplot(x = "pickup_community_area", y = "minutes_per_mile_avg", data = taxi_duration, color = "gray")			
165	予約語一覧の確認	-	-	# 予約語 __import__('keyword').kwlist	・予約語一覧の表示	松尾研	

166	組み込みオブジェクト一覧の表示	-	-	# 組み込みオブジェクト
				dir(__builtins__)
					・組み込みオブジェクト一覧の表示	松尾研	

167	format記法	-	strオブジェクト.format()	print('{0} と {1} を足すと {2}です'.format(2,3,5)) # ファイル名を機械的に生成する---------- # 分割してファイル書き出し for i in group: print('data{}.txt'.format(i), end=' ') res = data[data.group_id == i] **res.to_csv('data{}.txt'.format(i), sep='\t', index=False)**	・文字列埋め込み	松尾研	
168	Python3 文字列中に変数展開したい	Python3 文字列中に変数展開したい	Python3 文字列中に変数展開したい	**ex)#列名を動的に生成するのにも使える** # 良性（benign）クラスの予測確率が0.4、0.3、0.15、0.05以上なら、それぞれの列に1を設定する for threshold in [0.4, 0.3, 0.15, 0.05]: results[**'flag_%s' % threshold**] = results['benign'].map(lambda x: 1 if x > threshold else 0)	・文字列埋め込み	https://chaika.hatenablog.com/entry/	

id	library/module	category	name	usage	remarks	url	備考

169	辞書(dict)のforループ	辞書(dict)のforループ	辞書(dict)のforループ	# キーと値を取り出して表示する
				for key, value in dic_data**.items()**:
				　print(key, value)
				
				# キーだけを取り出す
				for key in dic_data**.keys()**:
				　print(key)
				
				# 値だけを取り出す
				for key in dic_data**.values():**
				　print(key)
				
				# キーと何かを組み合わせて取り出す
				from itertools import cycle
				colors = cycle(['red', 'aqua', 'darkorange', 'deeppink', 'cornflowerblue'])
				for model_name, color in **zip(models.keys(), colors):**
				　print(model_name, color)
						松尾研	

170	リスト内包表記	-	-	# リストを作る
				data_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
				
				# 空のリストを作成
				data_list1 = []
				
				# 内包表記、data_listから1つ1つ要素を取り出し、2倍した数字を新たな要素とするリストを作成
				data_list1 = [i * 2 for i in data_list]
				print(data_list1)
				
				#条件判断を含む内包表記
				[i * 2 for i in data_list if i %2 ==0]
				#[4, 8, 12, 16, 20]
				
				even_list = [i if i % 2 == 0 else "odd" for i in range(10)]
				
				#----------
				#npと普通の配列の計算速度の比較
				# Nは乱数の発生数、10の6乗
				N = 10**6
				
				# Python版 (以下のrange(N)は0からN-1までの整数を用意しています。
				# 「_」は、代入した値を参照しないときに使う慣例的な変数名です。
				# たとえば、for a in range(N)と書くのと同じですが、aと書くと、その値をあとで使うように見えるので、
				# その値を参照しないときは、for _ in range(N)のように慣例的に書く書き方です
				normal_data = [random.random() for _ in range(N)]
				
				# Numpy版
				numpy_random_data = np.array(normal_data)
				
				# calc time :合計値
				# ふつうの処理
				%timeit sum(normal_data)
				
				# Numpyを使った処理
				%timeit np.sum(numpy_random_data)
				
				#ジェネレータ内包表記
				def cipher(target):
				　result = ''.join(chr(219 - ord(c)) if c.islower() else c for c in target)
				　return result
					内包表記では、条件を指定し、条件に合致するものだけを新しいリストの対象にすることもできます。
					たとえば`data_list`から、値が偶数である要素だけを取り出したいときは、以下のようにします。「`if i %2 ==0`」の部分が指定している条件です。
					「`%`」は余りを計算する演算子です。つまり、`i % 2`は、`i`を2で割った余りです。これが0であるということは偶数であるということを示します。
						松尾研
						https://qiita.com/segavvy/items/55	

id	library/module	category	name	usage	remarks	url	備考
171	base	function	zip()	for one, two in zip([1, 2, 3], [11, 12, 13]): 　print(one, 'と', two) #1 と 11 #2 と 12 #3 と 13 ex2) # カイ2乗分布 # 自由度2, 10, 60に従うカイ二乗分布が生成する乱数のヒストグラム **#zip関数で[2, 10, 60]という配列と["b", "g", "r"]という配列から、[(2, "b"), (10, "g"), (60, "r")]というタプルの配列を作っている** **for df, c in zip([2, 10, 60], 'bgr'):** 　x = np.random.chisquare(df, 1000) 　plt.hist(x, 20, color=c) 　plt.grid(True) **# list(zip())の組み合わせはDataFrame作成に使える** feature_score = pd.DataFrame(**list(zip**(X_train.dtypes.index, catboost_model.get_feature_importance(Pool(X_train, label=Y_train, cat_features=cat_features)))), columns=['Feature','Score'])	`zip`関数は、それぞれ異なるリストを同時に取り出していく処理を実行します。たとえば、`[1,2,3]`というリストと、`[11,12,13]`という2つのリストがあるとき、それぞれ同じインデックスで値を取って表示したいとき――先頭の値である「1と11」、次の値である「2と12」、そして「3と13」のように繰り返して処理したいとき――は、次のようにします。 異なるリストデータがあったとして、それぞれのインデックスがお互いに対応していて、同時に取り出して処理したい場合は、zip関数は便利です。 ・複数の配列から、タプルの配列を作るために使うことも	松尾研	

172	while文	-	-	# 初期値の設定
				num = 1
				
				# while文の開始
				while num <= 10:
				　print(num)
				　num = num + 1
				# while文の終わり
				
				# 最後に代入された値を表示する
				print('最後の値は{0}です'.format(num))
				
				#途中で処理をストップさせたい場合----------
				# 初期値の設定
				num =1
				
				# while文の開始
				while num <= 10:
				　num = num + 1
				
				　# if文の開始
				　if num > 6:
				　　print('6を超えました')
				　　break
				　else:
				　　pass
				　# if文の終わり
				# while文の終わり
				
				print('最後の値は{0}です'.format(num))
				
				#ある条件で処理させ、ストップさせずにそのまま続けたい場合----------
				# 初期値の設定
				num =1
				
				# while文の開始
				while num <= 10:
				　num = num + 1
				
				　# if文の開始
				　if num > 6:
				　　print('6を超えました')
				　　continue
				　else:
				　　pass
				　# if文の終わり
				# while文の終わり
				
				print('最後の値は{0}です'.format(num))
					ある条件で処理をストップさせたい（while文を抜けたい）ときには、breakを使います。以下は、numの値が6を超えた時点で、whileの処理から抜け出して、終了しています。なお、if文のelseはなくてもよいですが、elseを書いて何も処理を書かないとエラーになるので、単にその個所をスキップするpassを書いています
					
					一方、ある条件で処理をさせて、ストップさせずにそのまま処理させたい（while文を続けたい）ときには、continueを使います。以下は、numの値が6を超えた時点で、'6を超えました'と表示されていますが、while文を抜けずに、そのまま処理を続けています		

id	library/module	category	name	usage	remarks	url	備考

173	class文	class文	class文	#クラス定義
				class MyCalcClass:
				　# 初期化
				　def __init__(self, x, y):
				　　self.x = x
				　　self.y = y
				　def calc_add1(self, a, b):
				　　return a + b
				　def calc_add2(self):
				　　return self.x + self.y
				　def calc_mutli(self, a, b):
				　　return a * b
				　def calc_print(self, a):
				　　print('data:{0}:yの値{1}'.format(a, self.y))
				
				#インスタンス作成
				instance_1 = MyCalcClass(1, 2)
				instance_2 = MyCalcClass(5, 10)
				
				#メソッド呼び出し
				print('2つの数の足し算(新たに数字を引数としてセット):', instance_1.calc_add1(5, 3))
				print('2つの数の足し算(インスタンス化の時の値):', instance_1.calc_add2())
				print('2つの数のかけ算:', instance_1.calc_mutli(5, 3))
				instance_1.calc_print(5)
					・クラスとしてMyCalcClassを作成しており、いくつかのメソッドを作っています。
					・インスタンスを生成するときは、クラスに実装した「__init__」という名前の特別なメソッドが実行されます。これを **コンストラクタ**と言います。コードには、「self.x = x」「self.y = y」という文があります。self.とは自分自身という意味です。そのため、この文によって、自身のx属性とy属性が、括弧のなかに指定した値になります。 つまり上記の例では、instanced_1では、MyCalcClass(1, 2)としているので、xが1、yが2となります。同様に、instance_2の場合はxが5、yが10となります。
						松尾研	

174	マジックコマンド	マジックコマンド	%quickref	%quickref	・マジックコマンドとは、Jupyter環境において、さまざまな環境操作をするための命令で、「%」から始まるコマンドです。デフォルトでは、「外部コマンドの実行（%run）」「ファイルのコピー（%cp）」「時間の計測（%time）」などの機能が用意されています。 ・標準のマジックコマンドは「ビルドインマジックコマンド」と呼ばれます。「%quickref」と入力して［Run］をクリックすると、一覧で表示できます。	松尾研	
175	マジックコマンド	マジックコマンド	%precision	%precision #小数点以下3位まで表示 %precision 3	・Numpyによる拡張です。データを表示する際に、小数、第何桁まで表示するのかを指定します。	松尾研	
176	マジックコマンド	マジックコマンド	%matplotlib	%matplotlib	・Matplotlibによる拡張です。グラフなどの表示方法を指定します。 「inline」と記述すると、その場所にグラフなどが表示されます。%matplotolibを指定しない場合は、別ウィンドウで表示されます。	松尾研	
177	built-in	function	time.time()	**ts = time.time()** matrix = [] cols = ['date_block_num','shop_id','item_id'] for i in range(34): 　sales = train[train.date_block_num==i] 　matrix.append(np.array(list(product([i], sales.shop_id.unique(), sales.item_id.unique())), dtype='int16')) **time.time() - ts**	・実行時間計測		
178	マジックコマンド	マジックコマンド	%time		・実行時間計測	https://ensekitt.hatenablog.com/entr	

179	マジックコマンド	マジックコマンド	%timeit	# Nは乱数の発生数、10の6乗
				N = 10**6
				
				# Python版 (以下のrange(N)は0からN-1までの整数を用意しています。
				# 「_」は、代入した値を参照しないときに使う慣例的な変数名です。
				# たとえば、for a in range(N)と書くのと同じですが、aと書くと、その値をあとで使うように見えるので、
				# その値を参照しないときは、for _ in range(N)のように慣例的に書く書き方です
				normal_data = [random.random() for _ in range(N)]
				
				# Numoy版
				numpy_random_data = np.array(normal_data)
				
				# calc time :合計値
				# ふつうの処理
				%timeit sum(normal_data)
				
				# Numpyを使った処理
				%timeit np.sum(numpy_random_data)
					・時間計測
					・100回同じ処理をして、ベスト3の平均計算時間を返すマジックコマンドです
					（Jupyter環境でRunを実行すると、100回実行されるのですから、その実行結果が表示されるまでには、しばらく時間がかかりますが、それは正常な動作です）。
					・たとえば、「100 loops, best of 3: 5.78 ms per loop」と表示されたときは、100回計算して、ベスト3の計算時間平均が5.78ミリ秒という意味です。
					・ 実行回数と平均回数は、それぞれnオプションとrオプションで変更できます。たとえば、「%timeit -n 10000 -r 5 sum(normal_data)」のようにすれば、1万回、ベスト5の平均計算時間という意味になります。なお、msはミリ秒で、μsはマイクロ秒（ミリ秒の1000分の1）です。
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

180	.dtype	プロパティ	.dtype	# 配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				# データの型
				data.dtype
				#dtype('int64')
					・型を調べるには、変数の後ろに「.dtype」のように指定します
					・「.dtype」という書き方は、「そのオブジェクトのdtypeプロパティを参照する」という意味
					　このようにピリオドで区切って、オブジェクトの状態を調べたり、オブジェクトが持つ機能（関数・メソッド・プロパティ）を実行したりするのは、オブジェクト型プログラミングの特徴
					
					・ちなみに、「.」を入力した後に Tabキーを押すと、その変数がもっているプロパティやメソッドの一覧が表示されるので、そこから該当のものを選ぶこともできます。そうすることで、すべてのプロパティやメソッドを正確に覚える必要がなくなり、タイプミスも減ります。
						松尾研	

181	.ndim	プロパティ	.ndim	# 配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				print('次元数:', data.ndim) #1
				print('要素数:', data.size) #10
					・配列の次元数と要素数を取得するには、それぞれ、ndimプロパティとsizeプロパティを参照します。これらのプロパティを確認すれば、データの大きさなどが、どのぐらいなのかがわかります。以下は次元数が1、要素数が10になっています。	松尾研	

182	.size	プロパティ	.size	# 配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				print('次元数:', data.ndim) #1
				print('要素数:', data.size) #10
					・配列の次元数と要素数を取得するには、それぞれ、ndimプロパティとsizeプロパティを参照します。これらのプロパティを確認すれば、データの大きさなどが、どのぐらいなのかがわかります。以下は次元数が1、要素数が10になっています。	松尾研	

183	numpy	method	sort()	# 配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				# ソートした結果を表示
				data.sort()
				print('ソート後：', data)
				
				#降順にソート
				data[::-1].sort()
				print('ソート後：', data)
					・デフォルトでは、昇順（小さい数字から大きい数字）になります
					・sortメソッドは、元のデータ（data）を置き換えるので注意しましょう。再度dataを表示すると、ソート後のデータになっているのがわかります。
					・降順（大きい数字から小さい数字）にしたい場合は、data[::-1].sort()のように、スライスを使って操作します。
						松尾研	

184	スライス	スライス	スライス		・ スライスはPythonの機能で、**[n:m:s]**のように記述すると、**「n番目からm-1番目を、sずつ飛ばして取り出す」**という意味になります。nやmを省略したときは「すべて」という意味になります。また**sが負のときは先頭からではなく、末尾から取り出すことを意味します** ・つまり、[::-1]は、「末尾から1つずつ取り出す」という意味になります	松尾研	

185	numpy	method	min()
			max()
			sum()
			cumsum()
				# 配列の作成
				data = np.array([9, 2, 3, 4, 10, 6, 7, 8, 1, 5])
				
				print('Min:', data.min())
				print('Max:', data.max())
				print('Sum:', data.sum())
				# 積み上げ
				print('Cum:', data.cumsum())
				# 積み上げ割合
				print('Ratio:', data.cumsum() / data.sum())
				
				#結果----------
				Min: 1
				Max: 10
				Sum: 55
				Cum: [10 19 27 34 40 45 49 52 54 55]
				Ratio: [0.182 0.345 0.491 0.618 0.727 0.818 0.891 0.945 0.982 1.]
				
				ex2)
				#累積和
				p = np.random.choice(sample_array, calc_times).cumsum()
				
				ex3)
				# arangeで9つの要素を持つ配列を生成。reshapeで3行3列の行列に再形成
				sample_multi_array_data1 = np.arange(9).reshape(3,3)
				
				print('最小値:',sample_multi_array_data1.min())
				print('最大値:',sample_multi_array_data1.max())
				print('平均:',sample_multi_array_data1.mean())
				print('合計:',sample_multi_array_data1.sum())
				
				# 行列を指定して合計値を求める
				print('行の合計:',sample_multi_array_data1.sum(axis=1))
				print('列の合計:',sample_multi_array_data1.sum(axis=0))
					・cumsumというメソッドは積上（前から順に足し上げていく）演算です。0番目の要素はそのまま、1番目の要素は0番目の要素+1番目の要素、2番目の要素は0番目の要素+1番目の要素+2番目の要素、…、という具合に足し上げたものです。	松尾研	

id	library/module	category	name	usage	remarks	url	備考

186	pandas	method	mean()
			median()
			mode()
			
			var()
			std()
				# 平均値
				print('平均値:', student_data_math['absences'].mean())
				# 中央値：中央値でデータを分けると中央値の前後でデータ数が同じになる（データの真ん中の値）、外れ値の値に影響を受けにくい
				print('中央値:', student_data_math['absences'].median())
				# 最頻値：最も頻度が多い値
				print('最頻値:', student_data_math['absences'].mode())
				#分散
				student_data_math['absences'].var()
				#標準偏差
				student_data_math['absences'].std()
				np.sqrt(student_data_math['absences'].var()) #これでも標準偏差が計算できる
					・平均値
					・中央値
					・最頻値
					・分散
					var()はデフォルトでは不偏分散(自由度n-1)を返す
					
					・標準偏差
						松尾研	

187	numpy	function	numpy.random.
			randn()
				import numpy.random as random
				
				random.seed(0)
				
				# 正規分布（平均0、分散1）の乱数を10個発生
				rnd_data = random.randn(10)
				
				print('乱数10個の配列:', rnd_data)
					・正規分布に従う乱数を発生させる	松尾研	

188	numpy	function	rand() random_sample() randint() randn() normal() binomial() beta() gamma() chisquare()	| 機能 | 意味 | |:---------|:-------| | `rand` | 一様分布。0.0以上、1.0未満 | | `random_sample` | 一様分布。0.0以上、1.0未満（`rand`とは引数の指定方法が異なる） | | `randint` | 一様分布。任意の範囲の整数 | | `randn` | 正規分布。平均0、標準偏差1の乱数 | | `normal` | 正規分布。任意の平均、標準偏差の乱数 | | `binomial` | 二項分布の乱数 | | `beta` | ベータ分布の乱数 | | `gamma` | ガンマ分布の乱数 | | `chisquare` | カイ二乗分布の乱数 |	・任意の分布に従う乱数を発生させる	松尾研	

189	二項分布を描く	二項分布を描く	二項分布を描く	# 二項分布
				np.random.seed(0)
				x = np.random.binomial(30, 0.5, 1000)
				plt.hist(x)
				plt.grid(True)
					・二項分布は、独立なベルヌーイ試行をn回繰り返したものです。
					・pythonでは、random.binomialを使って計算できます。
					・binomialに渡すパラメーターは先頭から順に、試行回数（n）、確率（p）、サンプル数です。
					・random.binomialはn回の試行のうち、確率pで生じる事象が発生する回数を返します
						松尾研	

190	ポワソン分布を描く	ポワソン分布を描く	ポワソン分布を描く	# ポアソン分布
				x = np.random.poisson(7, 1000)
				plt.hist(x)
				plt.grid(True)
					・1つ目のパラメータは、あの区間で事象が発生すると見込まれる回数で、ここでは7を設定しています。2つ目のパラメータはサンプル数です。	松尾研	

191	正規分布を描く	正規分布を描く	正規分布を描く	# 正規分布
				# np.random.normal(平均、標準偏差、サンプル数)
				x = np.random.normal(5, 10, 10000)
				plt.hist(x)
				plt.grid(True)
					・正規分布はガウス過程とも言われる	松尾研	

192	対数正規分布を描く	対数正規分布を描く	対数正規分布を描く	# 対数正規分布
				x = np.random.lognormal(30, 0.4, 1000)
				plt.hist(x)
				plt.grid(True)
					・対数正規分布は logxが正規分布に従うときの分布	松尾研	

193	カイ二乗分布を描く	カイ二乗分布を描く	カイ二乗分布を描く	# カイ2乗分布
				# 自由度2, 10, 60に従うカイ二乗分布が生成する乱数のヒストグラム
				for df, c in zip([2, 10, 60], 'bgr'):
				  x = np.random.chisquare(df, 1000)
				  plt.hist(x, 20, color=c)
				  plt.grid(True)
					・zip関数では[2, 10, 60]という配列と["b", "g", "r"]という配列から、[(2, "b"), (10, "g"), (60, "r")]というタプルの配列を作っている	松尾研	

194	t分布を描く	t分布を描く	t分布を描く	# t分布
				x = np.random.standard_t(5, 1000)
				plt.hist(x)
				plt.grid(True)
						松尾研	

195	F分布を描く	F分布を描く	F分布を描く	# F 分布
				for df, c in zip([(6, 7), (10, 10), (20, 25)], 'bgr'):
				  x = np.random.f(df[0], df[1], 1000)
				  plt.hist(x, 100, color=c)
				  plt.grid(True)
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

196	numpy	function	np.random.choice()	# 抽出対象データ
				data = np.array([9,2,3,4,10,6,7,8,1,5])
				
				# ランダム抽出
				# 10個を抽出（重複あり、復元抽出）
				print(random.choice(data, 10))
				# 10個を抽出（重複なし、非復元抽出）
				print(random.choice(data, 10, replace = False))
					・ランダムサンプリング
					・2つの引数と1つのオプションを指定します。1つ目の引数は、操作対象の配列、2つ目は取り出す数です。オプションはreplaceです。replaceをTrueにする、もしくは省略したときは、取り出すときに重複を許します。これを復元抽出と言います。replaceをFalseにしたときは、データの重複を許さずに取り出します。これを非復元抽出と言います。
						松尾研	

197	random	method	random.shuffle()	import random l = list(range(5)) print(l) # [0, 1, 2, 3, 4] random.shuffle(l) print(l) # [4, 3, 2, 1, 0]	・Shuffle list x in place, and return None. ・randomモジュールの関数shuffle()で、元のリストをランダムに並び替えられる。	https://note.nkmk.me/python-randc	

198	numpy	function	np.dot()	# データの準備
				array1 = np.arange(9).reshape(3,3)
				array2 = np.arange(9,18).reshape(3,3)
				
				# 行列の積
				np.dot(array1, array2)
				array1.dot(array2) #この書き方でも同じ
				
				# 要素どうしの積
				array1 * array2
					・行列の積
					・＊だと要素同士の積
						松尾研	

199	numpy	function	np.zeros()
			np.ones()
				#全ての要素が0の行列を作成
				print(np.zeros((2, 3), dtype = np.int64))
				
				#全ての要素が1の行列を作成
				print(np.ones((2, 3), dtype = np.float64))
					・dtypeオプション→データ型の指定
					int64→64ビット整数
					float64→64ビット浮動少数
						松尾研	

200	scipy	function	linalg.det()
			linalg.inv()
				# 線形代数用のライブラリ
				import scipy.linalg as linalg
				
				matrix = np.array([[1,-1,-1], [-1,1,-1], [-1,-1,1]])
				
				# 行列式
				print('行列式')
				print(linalg.det(matrix))
				
				# 逆行列
				print('逆行列')
				print(linalg.inv(matrix))
				
				#もとの行列を逆行列の積は単位行列
				print(matrix.dot(linalg.inv(matrix)))
					・行列式の計算
					・逆行列の計算
						松尾研	

201	scipy	function	linalg.eig()	# 線形代数用のライブラリ
				import scipy.linalg as linalg
				
				matrix = np.array([[1,-1,-1], [-1,1,-1], [-1,-1,1]])
				
				# 固有値と固有ベクトル
				eig_value, eig_vector = linalg.eig(matrix)
				
				# 固有値と固有ベクトル
				print('固有値')
				print(eig_value)
				print('固有ベクトル')
				print(eig_vector)
					・固有値の計算	松尾研	

id	library/module	category	name	usage	remarks	url	備考

202	pandas	function	pd.Series()
			
			.values
			.index
				import pandas as pd
				from pandas import Series, DataFrame
				
				# Series
				sample_pandas_data = pd.Series([0,10,20,30,40,50,60,70,80,90])
				print(sample_pandas_data)
				
				print('データの値:', **sample_pandas_data.values**)
				print('インデックスの値:', **sample_pandas_data.index**)
				
				# indexをアルファベットでつける
				sample_pandas_index_data = pd.Series(
				[0, 10,20,30,40,50,60,70,80,90],
				index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'])
				print(sample_pandas_index_data)
					・Seriesライブラリ
					一次元の配列を扱う
					
					・Seriesは1次元の配列のようなオブジェクト。PandasのベースはNumpyのarrayです
					・Seriesオブジェクトをprintすると、2つの組の値が表示されます。
					先頭の10行文は要素のインデックスと値です。dtypeはデータの型です。
					・データの値とインデックスの値は、**valuesプロパティ**と**indexプロパティ**を指定することで、別々に取り出すこともできます。
						松尾研	

203	pandas	function	pd.merge()	import pandas as pd
				from pandas import Series, DataFrame
				
				attri_data1 = {'ID':['100','101','102','103','104'],
				'City':['Tokyo','Osaka','Kyoto','Hokkaido','Tokyo'],
				'Birth_year':[1990,1989,1992,1997,1982],
				'Name':['Hiroshi','Akiko','Yuki','Satoru','Steve']}
				
				attri_data_frame1 = DataFrame(attri_data1)
				
				# データの列の削除
				attri_data_frame1.drop(['Birth_year'], axis = 1)
				
				# 別のデータの準備
				attri_data2 = {'ID':['100','101','102','105','107'],
				'Math':[50,43,33,76,98],
				'English':[90,30,20,50,30],
				'Sex':['M','F','F','M','M']}
				attri_data_frame2 = DataFrame(attri_data2)
				
				# データのマージ（内部結合、詳しくは次の章で）
				pd.merge(attri_data_frame1,attri_data_frame2)
				
				# データのマージ (内部結合)
				# キーは自動的に認識されるが、onで明示的に指定可能
				merge_data = pd.merge(student_data_math, student_por,
				on =
				['school','sex','age','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob','reason','nursery','internet'],
				suffixes=("_math", "_por")) #左のDFと右のDFのsuffixを指定するので2個ある
				
				# データのマージ（全結合）
				pd.merge(df1, df2, **how = 'outer'**)
				
				# index によるマージ
				pd.merge(df1, df2, left_index = True, right_on = 'index_num')
				
				# データのマージ（left）**left_join**
				pd.merge(df1, df2, **how = 'left'**)
				
				# 複数カラムでのjoin
				index_cols = ['shop_id', 'item_id', 'date_block_num']
				# Join it to the grid
				all_data = pd.merge(grid, gb, how='left', on=index_cols).fillna(0)
					・データフレームの結合
					・キーを明示しないときは、自動で同じキーの値であるものを見つけて(内部)結合する
					例の場合、キーはID
					・on→key列の指定
					・suffix→接尾字の指定 (mergeするDFに同じ名前のカラムがあった際に便利)
					
					・全結合
					どちらのデータにも存在するデータで結合する
					全結合ではhowパラメータにouterを指定
					結合する値がない場合は、NaNになります
					
					・indexによるマージ
					left_indexパラメータやright_onパラメータを使うと、キーをインデックスで指定して結合できます。
					例は、左側のデータのインデックスと、右側のデータのindex_numカラムをキーとして指定するものです
					
					・mergeすると、merge前のDFのindexが失われるので注意
					indexをdateにしていた場合などは困る
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

204	matplotlib	function	plt.plot()	# Matplotlib と Seabornの読み込み
				# Seabornはきれいに図示できる
				import matplotlib as mpl
				import seaborn as sns
				
				# pyplotにはpltの別名で実行できるようにする
				import matplotlib.pyplot as plt
				
				# Jupyter Notebook上でグラフを表示させるために必要なマジックコマンド
				%matplotlib inline
				
				# 散布図
				import numpy.random as random
				# シード値の固定
				random.seed(0)
				# x軸のデータ
				x = np.random.randn(30)
				# y軸のデータ
				y = np.sin(x) + np.random.randn(30)
				# グラフの大きさ指定（20や6を変更してみてください）
				plt.figure(figsize=(20, 6))
				# グラフの描写
				plt.plot(x, y, 'o')
				#以下でも散布図が描ける
				#plt.scatter(x, y)
				# タイトル
				plt.title('Title Name')
				# Xの座標名
				plt.xlabel('X')
				# Yの座標名
				plt.ylabel('Y')
				
				# grid（グラフの中にある縦線と横線）の表示
				plt.grid(True)
				
				ex2)
				# グラフを描画
				plt.plot(range(1,21), training_accuracy, label='Training')
				plt.plot(range(1,21), test_accuracy, label='Test')
				plt.ylabel('Accuracy')
				plt.xlabel('n_neighbors')
				plt.legend()
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

205	matplotlib	function	plt.plot()	# 連続曲線
				
				# シード値の指定
				np.random.seed(0)
				
				# データの範囲
				numpy_data_x = np.arange(1000)
				
				# 乱数の発生と積み上げ
				numpy_random_data_y = np.random.randn(1000).cumsum()
				
				# グラフの大きさを指定
				plt.figure(figsize=(20, 6))
				
				# label=とlegendでラベルをつけることが可能
				plt.plot(numpy_data_x, numpy_random_data_y, label='Label')
				plt.legend()
				
				plt.xlabel('X')
				plt.ylabel('Y')
				plt.grid(True)
				
				ex2) #3つのグラフを同じ図に描く
				# スプライン3次補間を計算してf2として追加する。パラメータに「cubic」を指定する
				f2 = interpolate.interp1d(x, y,'cubic')
				
				#曲線を出すために、xの値を細かくする。
				xnew = np.linspace(0, 10, num=30, endpoint=True)
				
				# グラフ化。fを直線で描き、f2を点線で描く
				plt.plot(x, y, 'o', xnew, f(xnew), '-', xnew, f2(xnew), '--')
				
				# 凡例
				plt.legend(['data', 'linear', 'cubic'], loc='best')
				plt.grid(True)
				
				ex2)#pandasのplot機能を使う場合
				df.plot(figsize = (15,6), legend = 'best', grid = True)
						松尾研	

206	matplotlib	グラフの分割	グラフの分割	# グラフの大きさを指定
				plt.figure(figsize=(20, 6))
				
				# 2行1列のグラフの1つ目
				plt.subplot(2,1,1)
				x = np.linspace(-10, 10,100)
				plt.plot(x, np.sin(x))
				
				# 2行1列のグラフの2つ目
				plt.subplot(2,1,2)
				y = np.linspace(-10, 10,100)
				plt.plot(y, np.sin(2*y))
				
				plt.grid(True)
						松尾研	

207	numpy	function	np.linespace()	# xとして、linspaceで、開始が0、終了が10、項目が11つの等間隔数列を生成
				x = np.linspace(0, 10, num=11, endpoint=True)
				# yの値を生成
				y = np.cos(-x**2/5.0)
				plt.plot(x,y,'o')
				plt.grid(True)
					・linspace(-10,10,100)は −10−10 から 1010 までの数を 100100 個に分割した数字リストを取り出す	松尾研	

208	numpy	function	np.random.uniform()	#0~1までの数を10個発生させる np.random.uniform(0, 1, 1000)	・一様乱数(ある数～別のある数まで等確率で発生する乱数)を発生させる	松尾研	
209	bash	bash	pwd	pwd	**「pwd」はPythonのプログラムではなく、シェルのコマンド**です。 **Jupyter環境では、ひとつのセルに「pwdなどのシェルのコマンド」と「Pythonのコマンド」を混ぜて書くことはできず、エラーとなるので注意**してください。	松尾研	

id	library/module	category	name	usage	remarks	url	備考

210	web上からzipファイルのダウンロードと保存			#ダウンロードファイルを置くディレクトリに予め移動しておく
				pwd
				mkdir chap3
				cd ./chap3
				
				# webからデータを取得したり、zipファイルを扱うためのライブラリ
				import requests, zipfile
				from io import StringIO
				import io
				
				# データがあるurlの指定
				url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/00356/student.zip'
				
				# データをurlから取得する
				r = requests.get(url, stream=True)
				
				# zipfileを読み込み展開する
				z = zipfile.ZipFile(io.BytesIO(r.content))
				z.extractall()
					Webからデータをダウンロードするには、requests.getを使います。このダウンロードしたデータを、io.BytesIOを使ってバイナリストリームとしてZipFileオブジェクトに与え、最後にextractall()を実行すると、ダウンロードしたZIP形式データを展開できます。ダウンロードが終了したら、データがちゃんとダウンロードされ、展開されているかチェックしましょう。lsコマンドを実行すると、カレントディレクトリのファイル一覧を表示できます	松尾研	

211	変動係数をまとめて計算			# 変動係数：欠席数
				student_data_math['absences'].std() / student_data_math['absences'].mean()
				
				# それぞれの変動係数をまとめて計算
				student_data_math.std() / student_data_math.mean()
				#DFの各カラムの変動係数が計算できる
						松尾研	

212	numpy	function	np.cov()	# 共分散行列
				np.cov(student_data_math['G1'], student_data_math['G3'])
				
				#array([[11.017, 12.188],
				#　　[12.188, 20.99]])
				#12.188が共分散
				#(1,1)はG1の分散
				#(2,2)はG3の分散
					共分散行列の計算	松尾研	

213	scipy	function	sp.stats.pearsonr()	sp.stats.pearsonr(student_data_math['G1'], student_data_math['G3']) #(0.801, 0.000) #相関係数は0.8、0.00はP値	ピアソン相関係数の計算	松尾研	

214	numpy	function	np.corrcoef()	# 相関行列
				np.corrcoef([student_data_math['G1'], student_data_math['G3']])
				#array([[1.　, 0.801],
				#　　[0.801, 1.　]])
				#自分自身の相関係数の部分は1になっている
					・相関係数行列の計算
					・それぞれの変数について、すべての組み合わせで相関係数を算出
						松尾研	

215	numpy	function	np.unique()	coin_data = np.array([0, 0, 0, 0, 0, 1, 1, 1]) np.unique(coin_data) #array([0, 1])	・一意な値の抽出 　Rのunique()と同じ	松尾研	
216	pandas	method	nunique()		pandas.Series.nunique(), pandas.DataFrame.nunique() ユニークな要素の個数をint, pandas.Seriesで返す ユニークな要素の値のリストをNumPy配列ndarrayで返す pandas.Series.value_counts() ユニークな要素の値とその出現回数をpandas.Seriesで返す pandas.Series.unique()	https://note.nkmk.me/python-panda	
217	パッケージのバージョンアップ	パッケージのバージョンアップ	パッケージのバージョンアップ	!pip install --upgrade matplotlib		松尾研	

218	numpy	function	np.append()	np.append()
				
				ex)
				data = np.array([])
				#np.array()だとエラーになる
				
				for i in range(0, 10):
				　x = np.random.normal(0, 1, 100).mean()
				　data = np.append(data, x)
				　#np.append(data, x)だけだと、data自身は置き換わらないので注意
				
				#ちなみに、appendを使わなくてもリスト内包表記でいける
				normal_sample_data = [np.random.normal(0, 1, 100).mean() for _ in range(N)]
					・ndarray末尾に要素を追加する	https://deepage.net/features/nump	

219	numpy	function	np.mgrid()	ex) x, y = np.mgrid[10:100:2, 10:100:2]	・画像などの座標位置とか3次元グラフのX,Y軸の座標とかのIndexに使われるmeshgridを生成する	松尾研 https://qiita.com/supersaiakujin/iter	
220	numpy	function	np.empty()		・配列の生成速度が若干高速になることがあるため、値を0や1で初期化する必要のない場合は、np.emptyを使おう	松尾研 https://deepage.net/features/nump	

id	library/module	category	name	usage	remarks	url	備考
221	numpy	function	np.linspace()	linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None) ex) def lik_func(x): return(x**3*(1-x)**2) x = np.linspace(0, 1, 100) plt.plot(x, lik_func(x))	・Return evenly spaced numbers over a specified interval. ・グラフ作成時に、x軸の値を作成するなど	松尾研	
222	scipy	function	stats.ttest_rel()	from scipy import stats t, p = stats.ttest_rel(student_data_merge.G1_math, student_data_merge.G1_por) print('p値 = ', p)	・t検定	松尾研	

223	numpy	function	np.copy()	sample_array = np.arange(10)
				
				sample_array_copy = np.copy(sample_array)
				print(sample_array_copy)
				
				sample_array_copy[0:3] = 20
				print(sample_array_copy)
				
				# 元のリストの要素は変更されていない
				print(sample_array)
					・参照ではなくてコピーしたいときは、次のようにcopyを使います。すると、コピーしたものを参照するようになるため、変更しても元のデータは影響を受けません	松尾研	

224	numpy	function	np.where()	# 条件制御のためのブールの配列を作成
				cond_data = np.array([True,True,False,False,True])
				
				# 配列x_arrayを作成
				x_array= np.array([1,2,3,4,5])
				
				# 配列y_arrayを作成
				y_array= np.array([100,200,300,400,500])
				
				# 条件制御実施
				print(np.where(cond_data,x_array,y_array))
					・条件によって採用するデータを切り分ける
					
					numpy.whereを使うと、2つのデータ XX とデータ YY があるとき、条件を満たすかどうかによって、XX の要素を取り出す、もしくは YY の要素を取り出すというように、取得するデータを切り分けられます。その書式は、次の通りです。
					
					numpy.where(条件の配列, Xのデータ, Yのデータ)
						松尾研	

225	numpy	function	np.exp()	np.exp(2) #e**2	・ネイピア指数関数	松尾研	

226	numpy	function	np.any()	# 真偽値の配列関数
				cond_data = np.array([True,True,False,False,True])
				
				print('Trueが少なくとも1つあるかどうか:',cond_data.any())
				#np.any(cond_data)でも同じ
				print('すべてTrueかどうか:',cond_data.all())
				#np.all(cond_data)でも同じ
						松尾研	

227	numpy	function	np.all()	# 真偽値の配列関数
				cond_data = np.array([True,True,False,False,True])
				
				print('Trueが少なくとも1つあるかどうか:',cond_data.any())
				#np.any(cond_data)でも同じ
				print('すべてTrueかどうか:',cond_data.all())
				#np.all(cond_data)でも同じ
						松尾研	

228	配列の操作	配列の操作	配列の操作	# データの準備
				sample_names = np.array(['a','b','c','d','a'])
				random.seed(0)
				data = random.randn(5,5)
				
				print(sample_names)
				print(data)
				
				#条件に合致する要素の抽出
				data[sample_names == "b"]
				
				#条件に合致する要素の個数
				sample_multi_array_data1 = np.arange(9).reshape(3,3)
				print(sample_multi_array_data1)
				print('5より大きい数字がいくつあるか:',(sample_multi_array_data1>5).sum())
						松尾研	

229	numpy	function	np.diag()	# 行列計算
				sample_multi_array_data1 = np.arange(9).reshape(3,3)
				print(sample_multi_array_data1)
				
				print('対角成分:',np.diag(sample_multi_array_data1))
				print('対角成分の和:',np.trace(sample_multi_array_data1))
					・対角成分の抽出	松尾研	

id	library/module	category	name	usage	remarks	url	備考

230	numpy	function	np.diag()	# 行列計算
				sample_multi_array_data1 = np.arange(9).reshape(3,3)
				print(sample_multi_array_data1)
				
				print('対角成分:',np.diag(sample_multi_array_data1))
				print('対角成分の和:',np.trace(sample_multi_array_data1))
					・対角成分の和	松尾研	

231	numpy	function	np.concentrate()	#　データの準備
				sample_array3 = np.array([[1,2,3],[4,5,6]])
				sample_array4 = np.array([[7,8,9],[10,11,12]])
				print(sample_array3)
				print(sample_array4)
				
				# 行方向に結合。パラメータのaxisに0を指定
				#bind_rows()と同じ
				np.concatenate([sample_array3,sample_array4],axis=0)
				
				# 列方向に結合
				#bind_cols()と同じ
				np.concatenate([sample_array3,sample_array4],axis=1)
					・配列の結合	松尾研	

232	numpy	function	np.vstack()	# vstackを使った行方向結合の方法
				np.vstack((sample_array3,sample_array4))
				
				ex2)----------
				print(a1)
				# [[1 1 1]
				# [1 1 1]]
				
				a2 = np.full((2, 3), 2)
				print(a2)
				# [[2 2 2]
				# [2 2 2]]
				
				print(np.vstack([a1, a2]))
				# [[1 1 1]
				# [1 1 1]
				# [2 2 2]
				# [2 2 2]]
					・配列の結合(行方向)
					　bind_rows()と同じ
					・括弧が2重でないとエラーになるので注意
						松尾研
						https://note.nkmk.me/python-nump	

233	numpy	function	np.hstack()	# 列方向結合の他の方法
				np.hstack((sample_array3,sample_array4))
					・配列の結合(列方向)
					　bind_cols()と同じ
					・括弧が2重でないとエラーになるので注意
						松尾研	

234	numpy	function	np.split()	# データの用意
				sample_array3 = np.array([[1,2,3],[4,5,6]])
				sample_array4 = np.array([[7,8,9],[10,11,12]])
				sample_array_vstack = np.vstack((sample_array3,sample_array4))
				# 作成したデータsample_array_vstackを表示
				sample_array_vstack
				
				# sample_array_vstackを3つに分割し、first、seocnd、thirdという3つの変数に代入
				first,second,third=np.split(sample_array_vstack,[1,3])
				
				# firstの表示
				print(first)
				# secondの表示
				print(second)
				# secondの最初の要素を取り出す
				second[0]
				# thirdの表示
				print(third)
					・配列の分割
					・例では、splitに[1, 3]というパラメータを指定しており、これが分割方法
					・具体的には~1（1の手前すべて）、1~3（1から3の手前のみ）、3~（3以降すべて）のインデックスで取り出すという意味
					・結果として、3つに分割される。インデックスは0から始まるという点に注意
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

235	built-in
	numpy
		function	repeat()
			np.repeat()
				# sample_array_vstack2を~2,2,3~4,5~の4つに分割し、first、second、third、fourthに代入する
				first,second,third,fourth=np.split(sample_array_vstack2,[2,3,5])
				print('・1つ目：\n',first,'\n')
				print('・2つ目：\n',second,'\n')
				print('・3つ目：\n',third,'\n')
				print('・4つ目：\n',fourth,'\n')
				
				# repeatを使うと、各要素が指定した回数だけ繰り返されて生成される
				first.repeat(5)
				
				ex2)
				import numpy as np
				group = np.arange(0, 12) #0~11までのarrayを作成
				np.repeat(group, 2)
				# array([0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11])
					・arrayから指定の回数だけ繰り返して要素を取り出す
					・numpy.repeatメソッドで同じ要素を繰り返して配列を作成できます。
						松尾研
						https://algorithm.joho.info/programi	

236	特異値分解	特異値分解	特異値分解	# (2,5)行列 A = np.array([[1,2,3,4,5],[6,7,8,9,10]]) # 特異値分解の関数linalg.svd U, s, Vs = sp.linalg.svd(A) m, n = A.shape S = sp.linalg.diagsvd(s,m,n) print('U.S.V* = \n',U@S@Vs) #@は行列の積を簡素化する演算子		松尾研	
237	文字列を逆順にする	文字列を逆順にする	文字列を逆順にする	words = "stressed" words_new = words[::-1] print(words_new)		https://redcat-prog.hatenadiary.org/e	
238	文字列を規則的に抽出する	文字列を規則的に抽出する	文字列を規則的に抽出する	text = "パタトクカシーー" text_new = text[0:7:2] #1文字目から7文字目まで2個飛ばしで抽出 print(text_new)			
239	文字列から先頭と末尾を除いた値を抽出する	文字列から先頭と末尾を除いた値を抽出する	文字列から先頭と末尾を除いた値を抽出する	word = "couldn't" word[1:-1] # ouldn	・word[-1]は末尾の文字列		

240	pandas	method	stack()	stack(level=-1, dropna=True)
				
				ex)
				# hier_dfを用意
				hier_df= DataFrame(
				  np.arange(9).reshape((3, 3)),
				  index = [
				  ['a', 'a', 'b'],
				  [1, 2, 2]
				  ],
				  columns = [
				  ['Osaka', 'Tokyo', 'Osaka'],
				  ['Blue','Red','Red']
				  ]
				)
				hier_df
				
				# ピボット操作で「Blue、Red」の列を行に変更
				hier_df.stack()
					・データのピボット
					  行と列を入れ替える
					・melt, gatherのような処理？
					・列を行にし、tidyデータに1歩近づけるような処理
					
					・indexの階層構造をflatにしたいだけなら、以下の方法もある
					DF.columns = ['col1', 'col2', ...]
					DF.reset_index(inplace = True)
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

241	pandas	method	unstack()	ex)
				# hier_dfを用意
				hier_df= DataFrame(
				　np.arange(9).reshape((3, 3)),
				　index = [
				　　['a', 'a', 'b'],
				　　[1, 2, 2]
				　],
				　columns = [
				　　['Osaka', 'Tokyo', 'Osaka'],
				　　['Blue','Red','Red']
				　]
				)
				hier_df
				
				# ピボット操作で「Blue、Red」の列を行に変更
				hier_df.stack()
				
				# unstackメソッドで、「Blue、Red」の行を列に変更
				hier_df.stack().unstack()
				
				ex2)
				student_data_math = pd.read_csv("student-mat.csv", sep=";")
				#年齢（age）×性別（sex）でG1の平均点を算出し、縦軸が年齢（age）、横軸が性別（sex）となるような表（テーブル）を作成
				student_data_math.groupby(["age", "sex"])["G1"].mean().unstack()
					・ピボットしたデータを元に戻す
					・spreadのような処理？
					・行を列にし、階層を1段深くしてtidyデータから遠ざけるような処理
					
					・indexの階層構造をflatにしたいだけなら、以下の方法もある
					DF.columns = ['col1', 'col2', ...]
					DF.reset_index(inplace = True)
						松尾研	

242	pandas.core.frame	method	reset_index()	reset_index(level=None, drop=False, inplace=False, col_level=0, col_fill='') ex) dat = pd.concat([train,test],sort=True).reset_index(drop=True) **ex2) indexの階層構造を取っ払って普通のDFにする例** group = train.groupby(['date_block_num','shop_id','item_id']).agg({'item_cnt_day': ['sum']}) group.columns = ['item_cnt_month'] group.reset_index(inplace=True)	・DFのindex(行名)を削除する ・デフォルトはdrop=FALSEで、index(行名)は列として追加される ・drop=TRUEを指定するとindexは列として追加されず、単に削除される For DataFrame with multi-level index, return new DataFrame with labeling information in the columns under the index names, defaulting to 'level_0', 'level_1', etc. if any are None. For a standard index, the index name will be used (if set), otherwise a default 'index' or 'level_0' (if 'index' is already taken) will be used.		

243	pandas	method	duplicated()	duplicated(subset=None, keep='first')
				
				ex)
				# 重複があるデータ
				dupli_data = DataFrame({
				　　'col1': [1, 1, 2, 3, 4, 4, 6, 6],
				　　'col2': ['a', 'b', 'b', 'b', 'c', 'c', 'b', 'b']
				})
				print('・元のデータ')
				dupli_data
				
				# 重複判定
				dupli_data.duplicated()
					・それぞれの行が確認され、重複があるときは、Trueとなります。ただし、重複のあるデータでも1回目ではFalseとなり、2回目からTrueになります。	松尾研	

244	pandas	method	drop_duplicates()	DataFrame.drop_duplicates(self, subset=None, keep='first', inplace=False)
				
				# 重複削除
				dupli_data.drop_duplicates()
				
				# 重複した行を除いたユニークなレコード数のカウント----------
				columns = ['KeywordId', 'AdGroupId', 'Device', 'Slot']
				test[columns].drop_duplicates().shape
				# (639360, 4)
					・重複したデータを削除した結果のデータが返される
					・デフォルトでは全カラムが重複判定に使われる
					　subsetを指定すると、重複判定に使うカラムの指定ができる
						https://pandas.pydata.org/pandas-	

id	library/module	category	name	usage	remarks	url	備考

245	built-in	function	map()	#とある関数
				def calc_double(x) :
				　return x * 2
				#for文で記述した場合
				for num in [1, 2, 3, 4]:
				　print(calc_double(num))
				
				#map関数を使うと、この処理をリストのまま処理でき、次のように書けます。
				list(map(calc_double, [1, 2, 3, 4]))
				#[2, 4, 6, 8]
				
				#このような書き方だと、別にcalc_double関数を定義しておかなければなりませんが、先に説明した無名関数を使うと、ここに直接関数の処理を記述でき、たとえば、次のように書けます。
				list(map(lambda x : x * 2, [1, 2, 3, 4]))
				
				ex2)#DFに無名関数を適用して列追加する場合
				# InvoiceNoの1文字目を抽出する処理。mapとLambda関数を使う
				online_retail_data_table['cancel_flg'] = online_retail_data_table.InvoiceNo.**map(lambda x:str(x)[0])**
				online_retail_data_table.groupby('cancel_flg').size()
				
				ex3)#DFに無名関数を適用して列追加する場合
				#　birth_year の上3つの数字・文字を取り出す
				df1['up_two_num'] = df1['birth_year'].map(lambda x: str(x)[0:3])
				df1
				
				#age列の値を2倍した列を末尾に追加
				student_data_math["age_double"] = student_data_math["age"].map(lambda x: x*2)
				student_data_math.head()
				
				# 良性（benign）クラスの予測確率が0.4、0.3、0.15、0.05以上なら、それぞれの列に1を設定する
				for threshold in [0.4, 0.3, 0.15, 0.05]:
				　results['flag_%s' % threshold] = results['benign'].map(lambda x: 1 if x > threshold else 0)
				
				ex4) #無名関数とmapの組み合わせによるカラム追加
				# 「fin_flg」カラムを追加し、もし「flg-50K」カラムの値が「>50K」だったら1、そうでなければ0をセットする
				adult['fin_flg'] = adult['flg-50K'].**map(lambda x: 1 if x ==' >50K' else 0)**
				adult.groupby('fin_flg').size()
				
				# 目的変数をフラグ化（0/1化）する
				mushroom_dummy['flg'] = mushroom['classes'].**map(lambda x: 1 if x =='p' else 0)**
					・map() はリストやタプルの各要素に指定した関数を適用し、各適用結果を順番に返すイテレータを返す関数
					・ map(func, iterator) の形で使う
					・map関数は、高階関数と呼ばれ、関数を引数や戻り値として使う関数で、各要素に対して、何か処理や操作したいときに使います。
					
					たとえば、次のように要素の値を2倍にして返す関数calc_doubleを定義するとします。
					
					・pandas.core.seriesにあるmapメソッドとは別物なので注意
						松尾研
						https://www.lifewithpython.com/20	

246	pandas	method	map()	**ex1) #excelのvlookup**
				# データ1の準備
				data1 = {
				　'id': ['100', '101', '102', '103', '104', '106', '108', '110', '111',' 113'],
				　'city': ['Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo', 'Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo'],
				　'birth_year': [1990, 1989, 1992, 1997, 1982, 1991, 1988, 1990, 1995, 1981],
				　'name': ['Hiroshi', 'Akiko', 'Yuki', 'Satoru', 'Steeve', 'Mituru', 'Aoi', 'Tarou', 'Suguru','Mitsuo']
				}
				df1 = DataFrame(data1)
				df1
				
				# 参照データ
				city_map ={
				　'Tokyo': 'Kanto',
				　'Hokkaido': 'Hokkaido',
				　'Osaka': 'Kansai',
				　'Kyoto':'Kansai'
				}
				city_map
				
				#　参照データを結合
				# df1のcityカラムをベースとして、上の参照データcity_mapから対応する地域名データを持ってきて、新しく一番右にregionというカラムとして追加する
				# もし対応するデータがなかったら、NaNになる。
				df1['region'] = df1['city'].**map(city_map)**
				df1
				
				ex2)# Kaggle coursera week3のスクリプトで使われていた一文
				# In our non-regularized case we just *map* the computed means to the `item_id`'s
				all_data['item_target_enc'] = all_data['item_id'].map(item_id_target_mean)
					・マッピング処理を行う
					・マッピング処理は、Excelのvlookup関数のような処理
					　共通のキーとなるデータに対して、一方の（参照）テーブルからそのキーに対応するデータを引っ張ってくる機能
						松尾研	DFへ列追加したい

id	library/module	category	name	usage	remarks	url	備考
247	pandas.core.series	method	map()	ex) shops['city'] = shops['shop_name'].str.split(' ').map(lambda x: x[0]) ex2) # if subtype is nan then type cats['subtype'] = cats['split'].map(lambda x: x[1].strip() if len(x) > 1 else x[0].strip())	・Seriesの各要素にマッピング処理を適用 ・引数は3種類 function, dict, or Series ・functionを引数とする際、ラムダ式のif文も使える ・DFの列追加に使える	https://pandas.py	DFへ列追加したい

248	pandas.core.frame	method	DF.apply()	apply(func, **axis=0**, broadcast=None, raw=False, reduce=None, result_type=None, args=(), **kwds)
				
				ex)
				sns.heatmap(cross_cluster_age.apply(lambda x : x/x.sum(), axis=1), cmap='Blues')
				
				ex)
				# 大きい数字_小さい数字でID連結する関数
				def merge_id(x):
				if x.FirstId >= x.SecondId:
				return x.FirstId_str + '_' + x.SecondId_str
				else:
				return x.SecondId_str + '_' + x.FirstId_str
				
				tmp['merge_id'] = tmp.apply(lambda x: merge_id(x), **axis=1**)
					・Apply a function along an axis of the DataFrame
					データフレームの任意の軸に対し、関数を適用する
					・pandas DFへ列追加するのにも使える
						https://qiita.com/
						https://pandas.py
							DFへ列追加したい

249	pandas.core.frame	method	df.assign()	assign(**kwargs)
				
				ex)
				test = df_train.query("name_count > 1") #自分以外に同じ名字の人がいる乗客を抽出
				test = test.assign(survive_rate_mod = lambda test: (test.survive_count - test.Survived)/(test.
				name_count - 1))
				
				ex2)#DFの文字列置換
				data.assign(open2 = lambda x: x.open.str.replace("$", ""))
				
				ex3)#一度に複数列増やす場合
				def replace_func(x):
				return(x.str.replace("$", ""))
				
				data = data.assign(open = replace_func(data.open),
				high = replace_func(data.high),
				low = replace_func(data.low),
				close = replace_func(data.close),
				next_weeks_open = replace_func(data.next_weeks_open),
				next_weeks_close = replace_func(data.next_weeks_close))
				
				ex4)
				auto = auto.assign(price=pd.to_numeric(auto.price))
				auto = auto.assign(horsepower=pd.to_numeric(auto.horsepower))
					・dplyr::mutate()のように新しく列追加する/列を書き換える際に使える
					・Assign new columns to a DataFrame, returning a new object (a copy) with the new columns added to the original ones.
					・Existing columns that are re-assigned will be overwritten.
						https://qiita.com/
						https://pandas.py
							DFへ列追加したい

id	library/module	category	name	usage	remarks	url	備考

250	pandas	function	pd.cut()	cut(x, bins, right=True, labels=None, retbins=False, precision=3, include_lowest=False,
				duplicates='raise')
				
				ex)
				# データ1の準備
				data1 = {
				'id': ['100', '101', '102', '103', '104', '106', '108', '110', '111',' 113'],
				'city': ['Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo', 'Tokyo', 'Osaka', 'Kyoto', 'Hokkaido', 'Tokyo'],
				'birth_year': [1990, 1989, 1992, 1997, 1982, 1991, 1988, 1990, 1995, 1981],
				'name': ['Hiroshi', 'Akiko', 'Yuki', 'Satoru', 'Steeve', 'Mituru', 'Aoi', 'Tarou', 'Suguru','Mitsuo']
				}
				df1 = DataFrame(data1)
				df1
				
				# 分割の粒度
				birth_year_bins = [1980, 1985, 1990, 1995, 2000]
				
				# ビン分割の実施
				birth_year_cut_data = pd.cut(df1.birth_year, birth_year_bins, right=True)
				birth_year_cut_data
				
				# 名前をつける場合
				# labelsパラメータを指定するとそれぞれのビンに名前をつけられる
				group_names = ['early1980s', 'late1980s', 'early1990s', 'late1990s']
				birth_year_cut_data = pd.cut(df1.birth_year, birth_year_bins, labels = group_names)
				pd.value_counts(birth_year_cut_data)
				
				# 数字で分割数指定も可能。ここでは2つに分割
				pd.cut(df1.birth_year, 2)
				
				ex2)
				# 分割のための区切りを設定
				bins = [15,20,25,30,35,40,45,50,55,60,65,100]
				
				# 上の区切りをもとに金融機関のデータを分割し、qcut_age変数に各データの年齢層を設定
				qcut_age = pd.cut(bank_with_cluster.age, bins, right=False)
				
				# クラスタ番号と年齢層を結合
				df = pd.concat([bank_with_cluster.cluster_number, qcut_age], axis=1)
					・ビン分割
					・Rのcut()と同じ
					・分位点での分割はpd.qcut()
					・right=Trueがデフォルトで、右側が閉区間
					right=Falseで右側が開区間
						松尾研	

251	pandas	function	pd.value_counts()	value_counts(values, sort=True, ascending=False, normalize=False, bins=None, dropna=True)
				
				# 集計結果
				pd.value_counts(birth_year_cut_data)
				
				ex2)
				# StockCodeごとに件数を数え、上位5件を表示
				trans['StockCode'].value_counts().head(5)
					・各値の数を集計する	松尾研	

252	pandas	function	pd.qcut()	#df1のbirth_year列の値を基準に、2分割する
				pd.value_counts(pd.qcut(df1.birth_year, 2)
					・pd.cut()の亜種
					・分位点での分割
					ほぼ同じサイズのビンを作成することができる
					・2番目の引数は分割する数
						松尾研	

253	pandas.core.frame	method	DF.shift()	shift(periods=1, freq=None, axis=0, fill_value=None)
				
				ex)
				import pandas_datareader.data as pdr
				start_date = '2001/1/2'
				end_date = '2016/12/30'
				fx_jpusdata = pdr.DataReader('DEXJPUS', 'fred', start_date, end_date)
				
				fx_jpusdata.resample('D').ffill().head()
				
				#データを1つ後にずらし、2001-01-02のレートは114.73でしたが、2001-01-03のレートとして扱われるように
				fx_jpusdata.shift(1).head()
				
				#前日のレートと当日のレートの比率を一気に出せる
				fx_jpusdata_ratio = fx_jpusdata / fx_jpusdata.shift(1)
				fx_jpusdata_ratio.head()
					・インデックスを固定したまま、データだけずらす
					Rでいうleadのような処理
						松尾研	

254	pandas.core.frame	method	DF.diff()			松尾研	
255	pandas.core.frame	method	DF.pct_change()			松尾研	

id	library/module	category	name	usage	remarks	url	備考

256	pandas.core.generic	method	DF.resample()	import pandas_datareader.data as pdr
				
				start_date = '2001/1/2'
				end_date = '2016/12/30'
				fx_jpusdata = pdr.DataReader('DEXJPUS', 'fred', start_date, end_date)
				
				#月末レートの抽出
				fx_jpusdata.resample('M').last().head()
				
				#日毎のデータの抽出
				fx_jpusdata.resample('D').last().head()
				
				#日毎のデータについて、欠損値を前の日の値で埋める
				fx_jpusdata.resample('D').ffill().head()
				
				#年単位で平均を取る
				fx_jpusdata.resample("A").mean()
				#fx_jpusdata.resample("Y").mean() これでも同じ
				
				# resample('B')でデータを営業日単位でリサンプリング。
				# ohlcメソッドで「open」「high」「low」「close」の4つのデータにする。
				df = pd.Series(rnd_walk, index=idx).resample('B').ohlc()
					・indexがdatetimeでないと使えない
					_DF.set_index()でindexを指定してから使う必要がある
					・resampleメソッドの引数にMを指定することで、月ごとのデータを取り出し、lastメソッドで末尾のデータを取り出しています。具体的には、以下の結果をみるとわかる通り、1月、2月、3月…の月末のレートを取り出せます。
					・日付を取り出したい場合は「D」、年を取り出したい場合は「Y」を、それぞれ引数に指定します。このように、ある頻度のデータを、別の頻度のデータで取り出し直す処理をリサンプリングといいます。また、最後のデータではなく、その平均を計算したい場合はmeanメソッドを使うことで計算できます
					・"M", "D"などのoffsetaliasesは以下URL参照
					http://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases
						松尾研	

257	pandas.core.generic	method	DF.rolling()	rolling(window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None) ex) import pandas_datareader.data as pdr start_date = '2001/1/2' end_date = '2016/12/30' fx_jpusdata = pdr.DataReader('DEXJPUS', 'fred', start_date, end_date) #3日間の移動平均を計算 fx_jpusdata.rolling(3).mean().head() #3日間の標準偏差を計算 fx_jpusdata.rolling(3).std().head()	・移動平均の計算	松尾研	

258	matplotlib	pyplot	plt.bar()	import matplotlib.pyplot as plt
				import matplotlib as mpl
				import seaborn as sns
				sns.set()
				%matplotlib inline
				
				# 表示するデータ
				x = [1, 2, 3]
				y = [10, 1, 4]
				
				# グラフの大きさ指定
				plt.figure(figsize = (10, 6))
				
				# align='center'でグラフを中央寄せ
				plt.bar(x, y, align='center', width = 0.5)
				
				# 棒グラフそれぞれのラベル
				plt.xticks(x, ['A Class', 'B Class', 'C Class'])
				
				# xとyのラベルを設定
				plt.xlabel('Class')
				plt.ylabel('Score')
				
				# グリッドを表示
				plt.grid(True)
					・棒グラフを描画
					・棒にラベルを表示したいときは、xtick関数を使って以下のように指定します。
					・そのまま実行するとグラフが左に寄ってあまり見栄えが良くないので、グラフを中央に寄せるために、align = 'center'パラメータを指定するとよい
					
					・なお、Pandasにも可視化機能が備わっており、plotメソッドでグラフ化できます。たとえばデータの後に「.plot(kind='bar')」と記すと、縦の棒グラフが描けます。「kind='barh'」にすれば横の棒グラフ、「kind='pie'」にすれば円グラフとなります。必要なときに使ってください。
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

259	matplotlib	pyplot	plt.bar()	# データの準備
				height1 = np.array([100, 200, 300, 400, 500])
				height2 = np.array([1000, 800, 600, 400, 200])
				
				# X軸
				x = np.array([1, 2, 3, 4, 5])
				
				# グラフの大きさ指定
				plt.figure(figsize = (10, 6))
				
				# グラフの描画
				p1 = plt.bar(x, height1, color = 'blue')
				p2 = plt.bar(x, height2, **bottom = height1**, color='lightblue')
				
				# 凡例を表示
				plt.legend((p1[0], p2[0]), ('Class 1', 'Class 2'))
				
				ex2)#pandasのplot機能を使う方法
				student_data_math.groupby("higher")["G3"].mean().plot(kind="bar")
				plt.xlabel("higher")
				plt.ylabel("G3 grade avg")
					・積み上げグラフを描画
					・同じくbar関数を使っていますが、bottomパラメータの設定に注目してください。上に積む方のグラフで、barのパラメータとしてbottom=＜下に積むグラフ＞を指定します。
						松尾研	

260	matplotlib	pyplot	plt.barh()	# 表示するデータ
				x = [1, 2, 3]
				y = [10, 1, 4]
				
				# グラフの大きさ指定
				plt.figure(figsize = (10, 6))
				
				plt.barh(x, y, align = 'center')
				# 棒グラフそれぞれのラベル
				plt.yticks(x, ['A Class','B Class','C Class'])
				plt.ylabel('Class')
				plt.xlabel('Score')
				plt.grid(True)
				
				ex2)#pandasのplot機能を使う方法
				student_data_math.groupby(["traveltime"])["G3"].mean().plot(kind="barh")
				plt.xlabel("G3 grade avg")
					・横向きの棒グラフを描画
					・xx の軸とyyの軸が入れかわるので、ラベルを再設定しています。
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

261	複数グラフの描画	複数グラフの描画	複数グラフの描画	# データの準備
				y1 = np.array([30, 10, 40])
				y2 = np.array([10, 50, 90])
				
				# X軸のデータ
				x = np.arange(len(y1))
				
				# グラフの幅
				w = 0.4
				
				# グラフの大きさ指定
				plt.figure(figsize = (10, 6))
				
				# グラフの描画。**y2の方はグラフの幅(wで指定している)の分、右にずらして描画する**
				plt.bar(x, y1, color = 'blue', width = w, label = 'Math first', align = 'center')
				plt.bar(x + w, y2, color='green', width = w, label = 'Math final', align = 'center')
				
				# 凡例を最適な位置に配置
				plt.legend(loc = 'best')
				
				plt.xticks(x + w / 2, ['Class A', 'Class B', 'Class C'])
				plt.grid(True)
				
				ex2)複数のヒストグラムの描画 (松尾研 DS講座 9章)
				# 乳がんデータを読み込むためのインポート
				from sklearn.datasets import load_breast_cancer
				
				# 乳がんデータの取得
				cancer = load_breast_cancer()
				
				# データをmalignant（悪性）かbenign（良性）に分けるためのフィルター処理
				# malignant（悪性）はcancer.targetが0
				malignant = cancer.data[cancer.target==0]
				
				# benign（良性）はcancer.targetが0
				benign = cancer.data[cancer.target==1]
				
				# malignant（悪性）がブルー、benign（良性）がオレンジのヒストグラム
				# 各図は、各々の説明変数（mean radiusなど）と目的変数との関係を示したヒストグラム
				fig, axes = plt.subplots(6,5,figsize=(20,20))
				ax = axes.ravel()
				for i in range(30):
				  _,bins = np.histogram(cancer.data[:,i], bins=50)
				  ax[i].hist(malignant[:,i], bins, alpha=.5)
				  ax[i].hist(benign[:,i], bins, alpha=.5)
				  ax[i].set_title(cancer.feature_names[i])
				  ax[i].set_yticks(())
				
				# ラベルの設定
				ax[0].set_ylabel('Count')
				ax[0].legend(['malignant','benign'],loc='best')
				fig.tight_layout()
						松尾研	

262	円グラフの描画	円グラフの描画	円グラフの描画	labels = ['Frogs', 'Hogs', 'Dogs', 'Logs']
				sizes = [15, 30, 45, 10]
				colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
				explode = (0, 0.1, 0, 0)
				
				# グラフの大きさ指定
				plt.figure(figsize = (15, 6))
				
				# グラフを表示
				plt.pie(sizes, explode = explode, labels = labels, colors = colors,
				  autopct = '%1.1f%%', shadow = True, startangle = 90)
				
				# 円を丸く描画
				plt.axis('equal')
				
				ex2)
				#pandasの描画機能を使う方法
				student_data_math.groupby("reason").size().plot(kind="pie", autopct="%1.1f%%")
					円グラフを描くにはpie関数を使って、それぞれのサイズやラベル等を設定します。axis関数で円グラフを丸く表示するように調整します。autopctパラメータでそれぞれの割合を表示する書式を指定します。またexplodeパラメータを指定すると、特定のカテゴリーだけ、円グラフの全体から離す調整ができます（ここでは、Hogsだけ0.1に設定しています）。
					
					startangleパラメータは、各要素の出力を開始する角度を表します。このパラメータを指定することで、出力開始位置を変更できます。「90」と指定すると上部中央が開始位置になり、反時計回りの方向に変更したい場合は正の値、時計回りに変更したい場合は負の値を指定します。
					
					出力する向きはcounterclockパラメータで指定します。Trueまたは指定しない場合は時計回りに、Falseと指定すると反時計回りに出力されます。
						松尾研	

id	library/module	category	name	usage	remarks	url	備考

263	バブルチャートの描画	バブルチャートの描画	バブルチャートの描画	N = 25
				
				# X,Yデータをランダムに生成
				x = np.random.rand(N)
				y = np.random.rand(N)
				
				# color番号
				colors = np.random.rand(N)
				
				# バブルの大きさをばらけさせる
				area = 10 * np.pi * (15 * np.random.rand(N)) ** 2
				
				# グラフの大きさ指定
				plt.figure(figsize = (15, 6))
				
				# グラフを描画
				plt.scatter(x, y, s = area, c = colors, alpha = 0.5)
				plt.grid(True)
					・要は散布図の1種	松尾研	

264	gzip	gzip	gzip.open()		・gzipファイルの読み込み	https://docs.python.org/ja/3/library/	

265	pandas	function	pd.data_range()	date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
				
				ex)
				# 日付データの設定。freq='T'で1分ごとにデータを生成する
				idx = pd.date_range('2015/01/01', '2015/12/31 23:59', freq='T')
						松尾研	

266	numpy	function	np.cumprod()	cumprod(a, axis=None, dtype=None, out=None) ex) a = np.array([1,2,3]) np.cumprod(a) #array([1, 2, 6])	・累積積の計算	松尾研	

267	pandas.core.groupby.groupby	method	DF.groupby().
			cumcount()
				>>> df = pd.DataFrame([['a'], ['a'], ['a'], ['b'], ['b'], ['a']],
				... columns=['A'])
				>>> df
				A
				0 a
				1 a
				2 a
				3 b
				4 b
				5 a
				>>> df.groupby('A').cumcount()
				0 0
				1 1
				2 2
				3 0
				4 1
				5 3
				dtype: int64
				>>> df.groupby('A').cumcount(ascending=False)
				0 3
				1 2
				2 1
				3 1
				4 0
				5 0
				dtype: int64
				
				ex2)
				test = pd.DataFrame({'item_id': [1, 1, 1, 2, 2, 2, 2], 'target': [3, 4, 5, 6, 7, 8, 9]})
				test
				
				test.groupby('item_id').cumcount()
					・Number each item in each group from 0 to the length of that group - 1
					・グループごとに連番を振る
						https://stackoverflow.com/questions/	

268	pandas.core.frame	method	DF.pivot()	pivot(index=None, columns=None, values=None) ex) #pivotを使う前にpivot用のindexとなるidをgroup毎に作っておく data["stock_group_id"] = data.groupby(["stock"]).cumcount() data[["stock", "close", "stock_group_id"]].head(15) data.pivot(index = "stock_group_id", columns = "stock", values = "close")	・indexに何も指定しないと、既存のindexを使うため、NaNが大量にできる	https://qiita.com/mwmsnn/items/6a	

id	library/module	category	name	usage	remarks	url	備考
269	seaborn	function	sns.heatmap()	plt.figure(figsize=(10, 7)) sns.heatmap(close_data_corr_by_stock) ex) sns.heatmap(cross_cluster_age.apply(lambda x : x/x.sum(), axis=1), cmap='Blues')	・ヒートマップを作成	https://note.nkmk.me/python-seabor	

270	pandasのplot機能まとめ	pandasのplot機能まとめ	pandasのplot機能まとめ	**ex1)#plotting multiple graphs on one plot**
				#1つのグラフ内に複数グラフをプロットする
				#CSCO, MSFTについて時系列グラフを描く
				plot_data = data[data["stock"].isin(["CSCO", "MSFT"])]
				plot_data.head()
				
				pd.pivot_table(data = plot_data, values = "close", columns = "stock", index = "date").plot(figsize = (10, 7), grid = True)
				
				#1つのグラフ内に複数グラフをプロットする ver2
				# クラスタリング結果のグラフ化
				ax = None
				colors = ['blue', 'red', 'green']
				for i, data in merge_data.groupby('cluster'):
				ax = data.plot.scatter(x='feature1', y='feature2', color=colors[i],
				label='cluster%d' % i, ax=ax)
				
				ex2)#棒グラフ
				# グラフを描画
				ax = labels.value_counts(sort=False).plot(kind='bar')
				ax.set_xlabel('cluster number')
				ax.set_ylabel('count')
				
				ex3)#複数グラフを並べてプロットする
				# resampleで時系列のデータを月別や四半期等に変更できる。今回は、月別(M)の合計を算出。そのあと、グラフ化
				top_five_country_data_country_totalP_index_uns.resample('M').sum()**.plot(subplots=True,figsize=(12,10))**
				
				# グラフが被らないように
				plt.tight_layout()
				
				ex3)#横向きの棒グラフ
				student_data_math.groupby(["traveltime"])["G3"].mean().plot(kind="barh")
				plt.xlabel("G3 grade avg")
					・複数グラフのプロット	https://stackoverflow.com/questions/	

271	numpy	function	np.log()	#1期前との対数時系列データ np.log(tmp/tmp.shift(1))	・対数を計算する	松尾研	
272	pandas	function	pd.read_excel()	pandas.read_excel(io, sheet_name=0, header=0, names=None, index_col=None, parse_cols=None, usecols=None, squeeze=False, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skiprows=None, nrows=None, na_values=None, keep_default_na=True, verbose=False, parse_dates=False, date_parser=None, thousands=None, comment=None, skip_footer=0, skipfooter=0, convert_float=True, mangle_dupe_cols=True, **kwds) ex) url = "http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx" excel_data = pd.read_excel(url, sheet_name="Online Retail")	・Read an Excel file into a pandas DataFrame ・excelファイルの読み込み	https://pandas.pydata.org/pandas-do	

273	pandas.core.strings	method	pd["column_name"].str.find()	find(sub, start=0, end=None)
				
				ex)
				#InvoiceNoで数字の前にCがあるものは除外する
				excel_data2 = excel_data2[excel_data2["InvoiceNo"].str.find("C").isnull()]
					・文字列検索、文字列マッチング	https://www.geeksforgeeks.org/pyth	

274	pandas	method	DF.set_index()	DataFrame.set_index(keys, drop=True, append=False, inplace=False, verify_integrity=False)
				
				ex)
				#InvoiceDateをindexにする
				excel_data2 = excel_data2.set_index("InvoiceDate")
				
				ex2)#indexを2つ指定することもできる
				# indexの設定（日時と国）
				top_five_country_data_country_totalP_index=top_five_country_data_country_totalP.set_index(['InvoiceDate','Country'])
				
				#indexを2つ指定してからunstackすると、2個目のindexがspreadされる
				top_five_country_data_country_totalP_index_uns = top_five_country_data_country_totalP_index.unstack()
					・Set the DataFrame index using existing columns.
					・カラムを使ってDFのindexを変える
						https://pandas.pydata.org/pandas-do	

id	library/module	category	name	usage	remarks	url	備考
275	pandas.core.series	method	Series.idxmax()	idxmax(axis=0, skipna=True, *args, **kwargs) ex) log_timeline.std().idxmax() #"CSCO"	・Return the row label of the maximum value ・最大値の行のラベルを返す 最大値をもつ行が複数ある場合は、最初の行を返す	松尾研	
276	pandas.core.series	method	Series.idxmin()	idxmin(axis=0, skipna=True, *args, **kwargs) ex) log_timeline.std().idxmin() #"KO"	・Return the row label of the minimum value ・最小値の行のラベルを返す 最小値をもつ行が複数ある場合は、最初の行を返す	松尾研	

277	pandas.core.series	method	Pandas Series.
			iteritems()
				#Seriesに1iteritems()メソッドを使うとイテレータになる
				test = 1
				for i in top_five_country_sales.Country.iteritems():
				　print(test)
				　print(i[1])
				　test += 1
					・pandas Seriesをイテレータに変換する
					※pandas Seriesはそのままではイテレータとして使えない
						https://www.geeksforgeeks.org/pyth	

278	複数グラフをループで作成	複数グラフをループで作成	複数グラフをループで作成	for i in top_five_country_sales.Country.iteritems():
				　#print(i[1])
				　plt.figure() #これがないと、グラフが1つしかできない
				　plot_data = top_five_country_description_top5_sales[top_five_country_description_top5_sales["Country"] == i[1]]
				　plt.pie(plot_data.sales, labels = plot_data.Description, counterclock=False, autopct='%.1f%%')
				　plt.ylabel(i[1])
				
				ex2)
				# クラスタリング結果のグラフ化
				# 1つのグラフに複数プロット
				ax = None
				colors = ['blue', 'red', 'green']
				for i, data in merge_data.groupby('cluster'):
				　ax = data.plot.scatter(x='feature1', y='feature2', color=colors[i],
				　　label='cluster%d' % i, ax=ax)
						松尾研	

279	pandas	function	pd.to_numeric()	auto = auto.assign(price=pd.to_numeric(auto.price)) auto = auto.assign(horsepower=pd.to_numeric(auto.horsepower)) print('データ型の確認（型変換後）\n{}'.format(auto.dtypes))	・数値型に変換する	松尾研	

280	sklearn	function	make_blobs()	# k-means法を使うためのインポート
				from sklearn.cluster import KMeans
				
				# データ取得のためのインポート
				from sklearn.datasets import make_blobs
				
				# サンプルデータ生成
				# 注意：make_blobsは2つの値を返すため、一方は使用しない「 _ 」で受け取る
				X, _ = make_blobs(random_state=10)
				
				# グラフを描画
				# colorのオプションで色付けができる
				plt.scatter(X[:,0],X[:,1],color='black')
					・make_blobs関数は縦軸と横軸に各々標準偏差1.0の正規分布に従う乱数を生成する関数で、主にクラスタリング用のサンプルデータ生成に使われます。
					・make_blobs関数には、とくに引数を与えなければ、-10から+10の範囲でランダムに2次元座標を選び、そこを中心に乱数の組を100個生成します。
						松尾研	

281	組み込み？	function	items()	for k, v in d.items(): 　print(k, v) # key1 1 # key2 2 # key3 3	・辞書型のオブジェクトをイテレータにする	https://note.nkmk.me/python-dict-ke	
282	クロス集計したい	クロス集計したい	クロス集計したい	ex) cross_cluster_job = bank_with_cluster.groupby(['cluster_number', 'job']).size().unstack().fillna(0)	・2軸でグループ化し、結果をunstack()するとクロス集計表のような見た目で結果が反る	松尾研	

id	library/module	category	name	usage	remarks	url	備考

283	built-in	function	set()
			issubset()
			intersection()
				# すべてのInvoiceNoをtrans_allとして抽出
				# 集合型とすることで、InvoiceNoを重複のない状態で保持できます
				# np.unique(trans.InvoiceNo)と同じ要素数
				trans_all = set(trans.InvoiceNo)
				
				# 商品85123Aを購入したデータをtrans_aとする
				trans_a = set(trans[trans['StockCode']=='85123A'].InvoiceNo) #1978
				print(len(trans_a))
				
				# 商品85099Bを購入したデータをtrans_bとする
				trans_b = set(trans[trans['StockCode']=='85099B'].InvoiceNo)
				print(len(trans_b))
				
				# 商品85123Aおよび85099Bを購入したデータをtrans_abとする
				# 積集合
				trans_ab = trans_a&trans_b
				print(len(trans_ab))
				trans_a.intersection(trans_b)
				
				# 商品組み合わせの支持度
				# trans_ab の、両商品を含むバスケットの数を表示
				print('両商品を含むバスケットの数:{}'.format(len(trans_ab)))
				print('両商品を含むバスケットの全体に占める割合:{:.3f}'.format(len(trans_ab)/len(trans_all)))
				
				# 商品自体の支持度
				print('商品85123Aのバスケットの数:{}'.format(len(trans_a)))
				print('商品85123Aを含むバスケットの全体に占める割合:{:.3f}'.format(len(trans_a)/len(trans_all)))
				
				# 確信度
				# 商品85123Aを購入するなら商品85099Bも購入する、というルールの確信度
				print('確信度:{:.3f}'.format(len(trans_ab)/len(trans_a)))
				
				#和集合
				X_set.union(Y_set)
					・setクラスのオブジェクトを作る
					・setは集合を扱うときに使い、重複のない要素をもつ順序なしのコレクションオブジェクトです。
					・積集合は、両方に共通するものを取り出すことで、setでは「&」を使います
					
					・部分集合か判定するならissubset()
						松尾研 9章
						https://note.nkmk
						https://docs.pyth
							集合演算をやりた

284	itertools	class function	itertools. combinations()	import itertools array = ['a', 'b', 'c'] result = list(itertools.combinations(array, 2)) print(result) # [('a', 'b'), ('a', 'c'), ('b', 'c')]	・組み合わせを作成する	http://pynote.hatenablog.com/entry/	

285	numpy	function	np.logspace()	scores = {}
				for gamma in np.logspace(-3, 2, num=6):
				  for C in np.logspace(-3, 2, num=6):
				  svm = SVC(gamma=gamma, C=C)
				  svm.fit(X_train,y_train)
				  scores[(gamma, C)] = svm.score(X_test, y_test)
					・np.linespaceの亜種
					・logspaceは、対数（底を省略したときは底は10）で指定した範囲の値を配列として生成します。この例では、10の-3乗から10の2乗の範囲を6等分した配列――具体的には、[0.001, 0.01, 0.1, 1, 10, 100]だけ繰り返します。
						松尾研	

286	numpy	function	ravel()	ravel(array, order='C')
				
				ex)
				# データの読み込み
				iris = datasets.load_iris()
				X = iris.data
				y = iris.target
				
				# 正解データのone-hot化
				y = label_binarize(y, classes=[0, 1, 2]) #01フラグ×3列に変形
				X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
				
				# multi-class classification model
				model = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True, random_state=0))
				y_score = model.fit(X_train, y_train).predict_proba(X_test)
				
				# 3つそれぞれのクラスについて、1次元のデータにして、ROC曲線、AUCを算出する
				# 75サンプル×3列→225サンプル×1列の形式にすれば、2クラス分類同様にROC曲線が描ける
				fpr, tpr, _ = roc_curve(y_test.**ravel()**, y_score.**ravel()**)
				roc_auc = auc(fpr, tpr)
					・Return a flattened array.
					・複数列のarrayをflattenして1列にする
					  1列にした分、行数がN(カラム数)-1倍増える
					・多クラス分類した結果(複数列)を1列にし、ROC曲線を描く際に使える
					・クラスラベルを複数列の01に変換するにはlabel_binarize()を使うと良い
						https://scikit-learn.org/stable/auto_e	

id	library/module	category	name	usage	remarks	url	備考

287	pandas	method	DF.join()	join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False)
				
				ex)
				# インポート
				from sklearn.datasets import load_boston
				
				# Housingデータセットを読み込み
				boston = load_boston()
				
				# DataFrameにデータを格納
				X = pd.DataFrame(boston.data, columns=boston.feature_names)
				
				# 住宅価格の中央値（MEDV）のデータを用意
				y = pd.Series(boston.target, name='MEDV')
				
				# Xとyを結合して先頭の5行を表示
				X.join(y).head()
					・DFに別のDFの列(Seriesも可)を結合する
					・merge()も同様		

288	numpy	function	np.squeeze()	squeeze(a, axis=None) ex) # 分析対象データ from sklearn.datasets import load_digits digits = load_digits() digits.images.shape # (1797, 8, 8) # サンプル毎に8行8列あるので、サンプル毎に1行64列にしたい **X = digits.images.reshape(1797, 1, 64).squeeze()** X.shape # (1797, 64)	・Remove single-dimensional entries from the shape of an array. 配列で1次元の部分を削除する ・配列のshapeを変えるreshapeと組み合わせて使うと便利	https://docs.scipy.org/doc/numpy/re	

289	pandas_profiling	function	pdp.ProfileReport()	# EDA
				import pandas_profiling as pdp
				pdp.ProfileReport(abalone_data)
				
				# HTMLで出力する場合
				profile = pdp.ProfileReport(abalone_data)
				profile.to_file("myoutputfile.html")
						https://qiita.com/h_kobayashi1125/it	

290	sklearn	function	label_binarize()	label_binarize(y, classes, neg_label=0, pos_label=1, sparse_output=False)
				
				y : array-like
				Sequence of integer labels or multilabel data to encode.
				
				classes : array-like of shape [n_classes]
				Uniquely holds the label for each class.
				
				neg_label : int (default: 0)
				Value with which negative labels must be encoded.
				
				pos_label : int (default: 1)
				Value with which positive labels must be encoded.
				
				sparse_output : boolean (default: False),
				Set to true if output binary array is desired in CSR sparse format
				
				ex)
				from sklearn import datasets
				from sklearn.preprocessing import label_binarize
				iris = datasets.load_iris()
				X = iris.data
				y = iris.target
				
				# Binarize the output
				y = label_binarize(y, classes=[0, 1, 2])
				n_classes = y.shape[1]
					・Binarize labels in a one-vs-all fashion
					クラスラベルを01×N列 形式に変換		

id	library/module	category	name	usage	remarks	url	備考
291	numpy	class function	np.c_	Examples | -------- | >>> np.c_[np.array([1,2,3]), np.array([4,5,6])] | array([[1, 4], | [2, 5], | [3, 6]]) | >>> np.c_[np.array([[1,2,3]]), 0, 0, np.array([[4,5,6]])] | array([[1, 2, 3, 0, 0, 4, 5, 6]]) ex2)---------- X_test_level2 = np.c_[pred_lr, pred_lgb]	・arrayをbind_colsする 配列を横に結合する		
292	sklearn	function	decision_function()	decision_function(self, X) ex) y_score = classifier.fit(X_train, y_train).decision_function(X_test)	Returns the distance of each sample from the decision boundary for each class. This can only be used with estimators which implement the decision_function method. ・それぞれのクラスラベルの境界線からの距離を返す	https://scikit-learn.org/stable/auto_e	
293	sklearn	function	roc_curve()	**roc_curve(y_true, y_score, pos_label=None, sample_weight=None, drop_intermediate=True)** y_true : array, shape = [n_samples] True binary labels. If labels are not either {-1, 1} or {0, 1}, then pos_label should be explicitly(はっきりと) given. y_score : array, shape = [n_samples] Target scores, can either be probability estimates of the positive class, confidence values, or non-thresholded measure of decisions (as returned by "decision_function" on some classifiers). ex)	・Compute Receiver operating characteristic (ROC) ROCを計算する ・Note: this implementation is restricted to the binary classification task 2値分類にしか適用できない ■引数 ・y_true 予測対象の実際の観測値 01や-1,1以外である場合、正例のラベルを指定する必要がある ・y_score 予測結果		
294	numpy	function	np.zeros_like()	zeros_like(a, dtype=None, order='K', subok=True)	・Return an array of zeros with the same shape and type as a given array. 与えられたarrayと同じshapeのarray of zeros を返す		
295	numpy	function	interp()	interp(x, xp, fp, left=None, right=None, period=None)	・与えられたxに対し、線形補間して計算したyを返す ・One-dimensional linear interpolation(補間). Returns the one-dimensional piecewise linear interpolant to a function with given discrete data points (`xp`, `fp`), evaluated at `x`.		
296	itertools	class function	cycle()	from itertools import cycle colors = cycle(['aqua', 'darkorange', 'cornflowerblue']) for i, color in zip(range(n_classes), colors): plt.plot(fpr[i], tpr[i], color=color, lw=lw, label='ROC curve of class {0} (area = {1:0.2f})' ".format(i, roc_auc[i]))			

297	matplotlib.pyplot	function	plt.subplots_adjust()	ubplots_adjust(left=None, bottom=None, right=None, top=None,
				wspace=None, hspace=None)
				
				ex)
				# Printing decision regions
				plt.subplot(3,2,j)
				plt.subplots_adjust(hspace = 0.4)
				plot_decision_regions(X = X_combined_sepal_standard
				, y = Y_combined_sepal
				, classifier = lr
				, test_idx = range(105,150))
				plt.xlabel('Sepal length')
				plt.ylabel('Sepal width')
				plt.title('C = %s'%i)
					・Tune the subplot layout	https://matplotlib.org/3.1.1/api/_as_	

id	library/module	category	name	usage	remarks	url	備考

298	numpy	function	np.meshgrid()	>>> import numpy as np
				>>> a = np.arange(0,3,0.5)
				>>> b = np.arange(0,10,2)
				>>> a
				array([0. , 0.5, 1. , 1.5, 2. , 2.5])
				>>> b
				array([0, 2, 4, 6, 8])
				>>> X, Y = np.meshgrid(a, b)
				>>> X
				array([[0. , 0.5, 1. , 1.5, 2. , 2.5],
				[0. , 0.5, 1. , 1.5, 2. , 2.5],
				[0. , 0.5, 1. , 1.5, 2. , 2.5],
				[0. , 0.5, 1. , 1.5, 2. , 2.5],
				[0. , 0.5, 1. , 1.5, 2. , 2.5]])
				>>> Y
				array([[0, 0, 0, 0, 0, 0],
				[2, 2, 2, 2, 2, 2],
				[4, 4, 4, 4, 4, 4],
				[6, 6, 6, 6, 6, 6],
				[8, 8, 8, 8, 8, 8]])
					・格子点の各座標を求めるのに便利	http://ailaby.com/contour/
						https://www.haya-programming.com
						https://deepage.net/features/numpy	

299	matplotlib	function	plt.contour()			https://matplotlib.org/3.1.0/api/_as_	
300	built-in	イテレータ	enumerate()	for i, name in enumerate(l): print(i, name) # 0 Alice # 1 Bob # 2 Charlie		https://note.nkmk.me/python-enume https://docs.python.org/ja/3/library/	
301	sklearn	function	validation_curve()	validation_curve(estimator, X, y, param_name, param_range, groups=None, cv='warn', scoring=None, n_jobs=None, pre_dispatch='all', verbose=0, error_score='raise-deprecating') ex)	・検証曲線(validation curve)を描く	https://scikit-learn.org/stable/module https://scikit-learn.org/stable/module	
302	warningを非表示にしたい					https://docs.python.org/ja/3/library/	
303	built-in	function	ord()	ord('a') # 97	・Return the Unicode code point for a one-character string.	https://docs.python.org/ja/3/library/	
304	built-in	function	chr()	chr(97) # a		https://docs.python.org/ja/3/library/	
305	built-in	function	str.upper() str.lower() str.capitalize() str.title() str.swapcase() str.isupper() str.islower() str.istitle()	str.upper(): すべての文字を大文字に変換 str.lower(): すべての文字を小文字に変換 str.capitalize(): 先頭の一文字を大文字、他を小文字に変換 str.title(): 単語の先頭の一文字を大文字、他を小文字に変換 str.swapcase(): 大文字を小文字に、小文字を大文字に変換 str.isupper(): すべての文字が大文字かどうか判定 str.islower(): すべての文字が小文字かどうか判定 str.istitle(): タイトルケースかどうか判定	・文字列判定関数	https://note.nkmk.me/python-capital	
306	ColabでGoogleドライブのファイルを読み込む	ColabでGoogleドライブのファイルを読み込む	ColabでGoogleドライブのファイルを読み込む		・URL参照 ・pwdでカレントディレクトリがわかる ・DF.to_csv()でcsv書き出しも普通にできる	https://qiita.com/uni-3/items/201aaa	

307	pandas	function	pandas.DataFrame.explode()	DataFrame.explode(self, column: Union[str, Tuple])
				
				ex)
				>>> df = pd.DataFrame({'A': [[1, 2, 3], 'foo', [], [3, 4]], 'B': 1})
				>>> df
				A B
				0 [1, 2, 3] 1
				1 foo 1
				2 [] 1
				3 [3, 4] 1
				
				>>> df.explode('A')
				A B
				0 1 1
				0 2 1
				0 3 1
				1 foo 1
				2 NaN 1
				3 3 1
				3 4 1
					・Transform each element of a list-like to a row, replicating the index values.	https://pandas.pydata.org/pandas-do	

id	library/module	category	name	usage	remarks	url	備考

308	math	function	math.ceil()	# mathモジュールをインポート
				import math
				# ここでも円周率を例にします
				a = 3.141592653589793
				#切り上げ
				math.ceil(a)
				#4
				#切り捨て
				math.floor(a)
				#3
					・小数点以下を切り上げる	https://hibiki-press.tech/learn_prog/	

309	jsonファイルを開きたい	jsonファイルを開きたい	jsonファイルを開きたい	# ファイルをColab上のディレクトリに置いてから解凍
				!gzip -d /content/jawiki-country.json.gz
				
				import json
				
				lines = []
				
				# ファイルを開いてファイルオブジェクトを返す
				with open('/content/jawiki-country.json', 'r') as data_file:
				  # イテレータで1行ずつ読み込む
				  for i, line in enumerate(data_file):
				  lines.append(line)
				  line_dict = json.loads(line)
				  if line_dict['title'] == 'イギリス':
				  print('----------イギリスは{0}番目の記事----------'.format(i))
				  print(line_dict['text'])
				
				# 実行結果から、イギリスの記事は先頭の記事を0番として、196番目の記事だとわかる			

310	キーワード	キーワード	in	print('hell' in 'hello kyoto') # => True print('heaven' in 'hello kyoto') # => False	・文字列の有無判定に使える	https://www.lifewithpython.com/20	

311	built-in	function	count()	>>> str = 'aAaAAbAccdd'
				
				# "A"という文字が何個あるか調べる
				>>> str.count('A')
				4
					count(...) method of builtins.str instance
					  S.count(sub[, start[, end]]) -> int
					
					  Return the number of non-overlapping occurrences of substring sub in
					  string S[start:end]. Optional arguments start and end are
					  interpreted as in slice notation.
						https://hibiki-press.tech/learn_prog/	

312	re	function	re.findall()	re.findall(pattern, string, flags=0)
				
				ex)
				regexp = '|.+ = .+\n'
				
				# 正規表現にマッチする部分をすべて抽出し、リスト形式で返す
				extract_list = re.findall(regexp, target_text)
					string 中の pattern による全ての重複しないマッチを、文字列のリストとして返します。string は左から右へ走査され、マッチは見つかった順で返されます。パターン中に 1 つ以上のグループがあれば、グループのリストを返します。パターンに複数のグループがあればタプルのリストになります。空マッチは結果に含まれます。	https://docs.python.org/ja/3/library/	

313	re	function	re.search()	re.search(pattern, string, flags=0)
				
				ex)
				# 正規表現でマッチした部分のみ抜き出し、整形する
				regexp = 'File:[^|]+||ファイル:[^|]+|'
				
				result = []
				for lines in uk_text:
				  search_res = re.search(regexp, lines)
				  if search_res:
				  match_file = re.sub('File:|ファイル:||', '', search_res[0])
				  result.append(match_file)
				  # print(match_file)
				
				print('\n'.join(result))
					string を走査し、正規表現 pattern がマッチを生じさせる最初の場所を探して、対応する マッチオブジェクト を返します。文字列内にパターンにマッチする場所がなければ None を返します。これは文字列のどこかで長さ 0 のマッチを見つけるのとは異なることに注意してください。	https://docs.python.org/ja/3/library/	

314	re	function	re.sub()	re.sub(pattern, repl, string, count=0, flags=0)	・正規表現でマッチした部分を置換する	https://docs.python.org/ja/3/library/	
315	pandas	method	Series.sort_index()		・pandasのSeriesをindexでソートする	https://pandas.pydata.org/pandas-do	

id	library/module	category	name	usage	remarks	url	備考

316	requests	method	get()	# imageinfoのサンプルスクリプトを実行
				# https://www.mediawiki.org/wiki/API:Imageinfo
				
				import requests
				
				S = requests.Session()
				
				URL = "https://en.wikipedia.org/w/api.php"
				
				PARAMS = {
				"action": "query",
				"format": "json",
				"prop": "imageinfo",
				"titles": "File:Billy_Tipton.jpg",
				"iiprop": "user|mediatype|url" # iiprop：which file information to get
				}
				
				R = S.get(url=URL, params=PARAMS)
				
				# urlの確認
				R.url
				
				DATA = R.json()
				
				DATE
					・HTTPでGETリクエストを送信
					・urlのパラメタはparams引数で渡す
						https://requests-docs-ja.readthedo
						https://www.mediawiki.org/wiki/API	

317	MeCab	method	parse()	import MeCab mecab_tagger = MeCab.Tagger() print(mecab_tagger.parse("すももももももものうち")) すもも　　名詞,一般,*,*,*,*,すもも,スモモ,スモモ も　　助詞,係助詞,*,*,*,*,も,モ,モ もも　　名詞,一般,*,*,*,*,もも,モモ,モモ も　　助詞,係助詞,*,*,*,*,も,モ,モ もも　　名詞,一般,*,*,*,*,もも,モモ,モモ の　　助詞,連体化,*,*,*,*,の,ノ,ノ うち　　名詞,非自立,副詞可能,*,*,*,うち,ウチ,ウチ EOS	・MeCab.Tagger()クラスのオブジェクトを使い、形態素解析する	https://qiita.com/menon/items/f041	
318	built-in	method	splitlines()	text = "吾輩は猫である。" mecab_tagger.parse(text).splitlines() ['吾輩\t名詞,代名詞,一般,*,*,*,吾輩,ワガハイ,ワガハイ', 'は\t助詞,係助詞,*,*,*,*,は,ハ,ワ', '猫\t名詞,一般,*,*,*,*,猫,ネコ,ネコ', 'で\t助動詞,*,*,*,特殊・ダ,連用形,だ,デ,デ', 'ある\t助動詞,*,*,*,五段・ラ行アル,基本形,ある,アル,アル', '。\t記号,句点,*,*,*,*,。,。,。', 'EOS'] mecab_tagger.parse(text).splitlines()[:-1] とすると、末尾の「EOS」は除外できる	・文字列を改行で分割する	https://note.nkmk.me/python-split-rs https://docs.python.org/ja/3/library/s	

319	datetime	function	datetime.date()	import datetime
				
				datetime.date(2019, 9, 15)
				
				# Athenaの出力結果を日付で絞る場合
				output = query_res[query_res["date"] == datetime.date(2019, 9, 11)]
					・datetime型の変数を作成	https://qiita.com/motoki1990/items/	

320	collections	class function	Counter() update() most_common()	words = Counter(['a', 'a', 'b', 'c', 'c', 'a']) words # Counter({'a': 3, 'b': 1, 'c': 2}) **# dict型に共通のメソッド(update)により、要素追加が可能** words.**update**(["a", "a", "b"]) words # Counter({'a': 5, 'b': 2, 'c': 2}) **# カウントの多い要素順に表示** words.**most_common()**	・単語の出現頻度を数える際に便利 ・リストを引数にとる	https://kaworu.jpn.org/kaworu/201	

id	library/module	category	name	usage	remarks	url	備考

321	pandas	method	rank()	ex)
				# 単語の出現頻度
				freq_df = pd.DataFrame({"frequency": [element[1] for element in words.most_common()]})
				
				# 出現頻度ランキングの付与
				freq_df["ranking"] = freq_df.rank(ascending=False, method="min")
					・順次付けを行う	https://note.nkmk.me/python-panda
						前処理大全P80	

322	両対数グラフを描きたい	両対数グラフを描きたい	両対数グラフを描きたい	# 両対数グラフ
				plt.figure(figsize=(10, 6))
				plt.plot(freq_df.frequency, freq_df.ranking)
				
				ax = plt.gca()
				
				ax.set_yscale('log')
				ax.set_xscale('log')
				
				plt.grid(which="both")
				
				plt.title("frequency and rank")
				plt.xlabel('ranking',fontsize=18)
				plt.ylabel('frequency',fontsize=18)
						https://qiita.com/sci_Haru/items/68b	

323	assert文	assert文	assert文		条件式がTrueではない時に、例外を投げます。 これを仕込んでおくと、それまでちゃんと動いていたコードが、いじっているうちにいつの間にか想定と異なる振る舞いをするようになった時に、いち早く気づくことが出来ます。「とにかく想定と違ったら止める」	https://qiita.com/nannoki/items/150	

324	scipy.sparse	function/method	lil_matrix()
			lil_matrix_obj.tocoo()
			lil_matrix_obj.todense()
			
			coo_matrix()
			coo_matrix_obj.max()
			coo_matrix_obj.sum()
				from scipy.sparse import csr_matrix, csc_matrix, coo_matrix, lil_matrix
				
				# 5×5の疎行列を作成
				mat_tmp = lil_matrix((5, 5))
				
				# 非ゼロ要素を設定する
				for i in range(0, 5):
				  for j in range(0, 5):
				  if i == j:
				  mat_tmp[i, j] = 1
				
				print(mat_tmp)
				
				# coo matrixに変換
				mat_coo = mat_tmp.tocoo(copy = True)
				
				# 最大の要素を確認
				mat_coo.max()
				mat_coo.sum()
					・sparse matrix (疎行列)の生成と演算
					・最後のリンク先がわかりやすい
						https://note.nkmk.me/python-scipy-
						http://hamukazu.com/2014/09/26/s
						https://docs.scipy.org/doc/scipy/ref	

325	疎行列から行指定で行列を抽出(スライス)したい	疎行列から行指定で行列を抽出(スライス)したい	疎行列から行指定で行列を抽出(スライス)したい	# sparse matrixのスライス
				# csrでないとうまくスライスできないので注意
				
				# 疎行列の作成
				mat_tmp = lil_matrix((5, 5))
				# 非ゼロ要素を設定
				mat_tmp[0, 0] = 1
				mat_tmp[0, 3] = 1
				mat_tmp[2, 4] = 1
				mat_tmp = mat_tmp.tocoo(copy = True) # coo matrixに変換
				# csrへの変換
				mat_tmp = mat_tmp.tocsr()
				
				# 普通のmatrixに変換してprintで確認
				print(mat_tmp.todense())
				# [[1. 0. 0. 1. 0.]
				# [0. 1. 0. 0. 0.]
				# [0. 0. 1. 0. 1.]
				# [0. 0. 0. 1. 0.]
				# [0. 0. 0. 0. 1.]]
				
				# スライスで0行目、2行目、2行目を抜き出したもの
				print(mat_tmp[np.array([0, 2, 2])].todense())
				# [[1. 0. 0. 1. 0.]
				# [0. 0. 1. 0. 1.]
				# [0. 0. 1. 0. 1.]]
					・sparse matrix (疎行列)のスライス	https://cmdlinetips.com/2019/07/hc	

id	library/module	category	name	usage	remarks	url	備考

326	疎行列で内積を計算したい	疎行列で内積を計算したい	疎行列で内積を計算したい	mat_a = mat_tmp = lil_matrix((3, 2))
				mat_b = mat_tmp = lil_matrix((1, 2))
				
				mat_a[0,0] = 1; mat_a[0,1] = 2; mat_a[1,0] = -1; mat_a[2,0] = 3; mat_a[2,1] = 1
				mat_b[0,0] = 7; mat_b[0,1] = 1
				
				print(mat_a.todense())
				print(mat_b.todense())
				
				mat_a.shape
				mat_b.shape
				
				mat_a = mat_a.tocsr()
				mat_b = mat_b.tocsr()
				
				# 内積の計算
				mat_c = mat_a.**multiply**(mat_b)
				print(mat_c.todense())
				
				mat_c.**sum**(axis=1)
					・sparse matrix (疎行列)の内積(dot product)
					・multiplyはpoint wise multiplicationなので、内積を計算する場合はmultiplyの後にsum(axis = 1)を使う
						https://docs.scipy.org/doc/scipy/ref
						https://note.nkmk.me/python-scipy-	

327	matrixをnp.arrayに変換したい	matrixをnp.arrayに変換したい	matrixをnp.arrayに変換したい	# fがnumpy.matrixlib.defmatrix.matrixだとする **np.array(f).ravel()** # f.reshape(1, f.size)		https://qiita.com/itoru257/items/dc0	

328	itertools	function	product()	import itertools
				import pprint
				
				l1 = ['a', 'b', 'c']
				l2 = ['X', 'Y', 'Z']
				
				p = itertools.product(l1, l2)
				list(p)
				
				ex2)----------
				from itertools import product
				
				ts = time.time()
				matrix = []
				cols = ['date_block_num','shop_id','item_id']
				for i in range(34):
				  sales = train[train.date_block_num==i]
				  # itertoolsのproduct()で要素同士の全ての組み合わせを返す
				  matrix.append(np.array(list(**product**([i], sales.shop_id.unique(), sales.item_id.unique())),
				dtype='int16'))
				
				time.time() - ts
				
				matrix = pd.DataFrame(**np.vstack**(matrix), columns=cols)
					・Pythonで複数のリストの直積（デカルト積）を生成するにはitertools.product()を使う。
					・2つのリストから1つずつ要素を取り出してペアとする際の、全ての組み合わせを返す
					・product()とnp.vstack()をあわせて使うことが多い
						https://note.nkmk.me/python-itertoo	

329	pandas	method	clip()	DataFrame.clip(self, lower=None, upper=None, axis=None, inplace=False, *args, **kwargs)
				
				ex)
				matrix['item_cnt_month'] = (matrix['item_cnt_month']
				  .fillna(0)
				  .**clip(0,20)** # NB clip target here
				  .astype(np.float16))
					Trim values at input threshold(s).
					Assigns values outside boundary to boundary values. Thresholds can be singular values or array like, and in the latter case the clipping is performed element-wise in the specified axis.
					
					・上限、下限を定めて、はみ出す値は上限値or下限値に置き換える
					clip→刈る
						https://pandas.pydata.org/pandas-do	

330	pandas.core.frame	method	iterrows()	ex)
				for idx, row in matrix.iterrows():
				  key = str(row.item_id)+' '+str(row.shop_id)
				  if key not in cache:
				  if row.item_cnt_month!=0:
				  cache[key] = row.date_block_num
				  else:
				  last_date_block_num = cache[key]
				  matrix.at[idx, 'item_shop_last_sale'] = row.date_block_num - last_date_block_num
				  cache[key] = row.date_block_num
				
				# イテレータの返す値の最初の1個を取り出す。(確認に便利
				next(matrix.iterrows())
					・Iterate over DataFrame rows as (index, Series) pairs.
					行番号、行のSeries をイテレータで返す
						https://pandas.pydata.org/pandas-do	

331	pandas.core.frame	method	melt()		・Rのmeltと同じ		

id	library/module	category	name	usage	remarks	url	備考

332	pandas.Series	method	rolling()	Series.rolling(self, window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None)
				
				ex)
				plt.figure(figsize=(16,6))
				plt.plot(ts.rolling(window=12,center=False).mean(),label='Rolling Mean'); #tsはpandas.Series
				plt.plot(ts.rolling(window=12,center=False).std(),label='Rolling sd');
				plt.legend();
				
				ex2)
				# groupbyとrollingの組み合わせ
				f_max = lambda x: x.rolling(window=3, min_periods=1).max() #起点月含む過去3ヶ月の最大値
				train_monthly[('item_cnt_%s' % 'max')] = train_monthly.sort_values('date_block_num').groupby
				(['shop_id', 'item_category_id', 'item_id'])['item_cnt'].apply(f_max)
				※列追加するだけでちゃんとmappingできてないんんじゃないかと思いきや、ちゃんとmappingできている
					・Provide rolling window calculations.
					移動平均を計算する
						https://pandas.pydata.org/pandas-
						https://www.kaggle.com/dimitreoliv	

333	statsmodels.tsa.seasonal	function	seasonal_decompose()	seasonal_decompose(x, model='additive', filt=None, freq=None, two_sided=True)
				
				ex)
				import statsmodels.api as sm
				# multiplicative
				res = sm.tsa.seasonal_decompose(ts.values,freq=12,model="multiplicative") #model="multiplicative"
				#plt.figure(figsize=(16,12))
				fig = res.plot()
				#fig.show()
					・Seasonal decomposition using moving averages
					・**model = 'additive'**
					→yt=St+Tt+Et
					→季節成分と残差をトレンドに加えると原系列になるよう分解
					
					・**model = 'multiplicative'**
					→yt=St x Tt x Et
					→季節成分と残差をトレンドに掛けると(？)原点系列になるよう分解？
						https://www.statsmodels.org/stable
						
						https://www.kaggle.com/jagangupt	

334	数値データのメモリ使用量を減らしたい
	float64→float32
	int64→int32
		数値データのメモリ使用量を減らしたい
		float64→float32
		int64→int32
			数値データのメモリ使用量を減らしたい
			float64→float32
			int64→int32
				def downcast_dtypes(df):
				'''
				Changes column types in the dataframe:
				
				`float64` type to `float32`
				`int64` type to `int32`
				'''
				
				# Select columns to downcast
				float_cols = [c for c in df if df[c].dtype == "float64"]
				int_cols = [c for c in df if df[c].dtype == "int64"]
				
				# Downcast
				df[float_cols] = df[float_cols].astype(np.float32)
				df[int_cols] = df[int_cols].astype(np.int32)
				
				return df
					・関数を作成する	Kaggle coursera	

335	数値データのメモリ使用量を減らしたい
	float64→float32
	int64→int32
		数値データのメモリ使用量を減らしたい
		float64→float32
		int64→int32
			数値データのメモリ使用量を減らしたい
			float64→float32
			int64→int32
				# Integer features (used by catboost model).
				int_features = ['shop_id', 'item_id', 'year', 'month']
				
				X_train[int_features] = X_train[int_features].astype('int32')
				X_validation[int_features] = X_validation[int_features].astype('int32')
					・数値データの列名のリストを作ってまとめて処理	https://www.kaggle.com/dimitreolive	

336	gc	function	gc.collect()	# Downcast dtypes from 64 to 32 bit to save memory
				all_data = downcast_dtypes(all_data)
				del grid, gb
				gc.collect();
					・メモリ解放を明示的に指定する？	https://docs.python.org/ja/3/library/	

337	pandas.core.index.base	method	difference()	# all_data(DF)の列名のうち、index_cols(リスト)に含まれない列名を返す
				all_data.columns.difference(index_cols)
					・残った列名を取得したい場合	https://pandas.pydata.org/pandas-do	

338	tqdm	function	tqdm()	from tqdm import tqdm_notebook as tqdm #Jupyter で使う場合はこっち # from tqdm import tqdm shift_range = [1, 2, 3, 4, 5, 12] for month_shift in tqdm(shift_range): print(month_shift)	・プログレスバーを表示させる ・イテラブルな物であれば何でも渡せる	https://blog.amedama.jp/entry/2018	
339	指定の列名だけ規則的に名前変更したい	指定の列名だけ規則的に名前変更したい	指定の列名だけ規則的に名前変更したい	index_cols = ['shop_id', 'item_id', 'date_block_num'] foo = lambda x: '{}_lag_{}'.format(x, month_shift) if x in cols_to_rename else x train_shift = train_shift.rename(columns=foo)		Kaggle coursera	
340	numpy	function	np.isclose()	ex) np.all(np.isclose(X_train_level2.mean(axis=0), [1.50148988, 1.38811989]))	・Returns a boolean array where two arrays are element-wise equal within a tolerance.	https://numpy.org/devdocs/referenc	
341	numpy	function	np.all()	ex) np.all(np.isclose(X_train_level2.mean(axis=0), [1.50148988, 1.38811989]))	・Test whether all array elements along a given axis evaluate to True.	https://numpy.org/devdocs/referenc	

id	library/module	category	name	usage	remarks	url	備考
342	numpy.arrayの任意の場所へ値(ベクトル)を代入したい	numpy.arrayの任意の場所へ値(ベクトル)を代入したい	numpy.arrayの任意の場所へ値(ベクトル)を代入したい	ex) X_train_level2_tmp[:, 0][dates_train_level2 == cur_block_num] = pred_lr X_train_level2_tmp[:, 1][dates_train_level2 == cur_block_num] = pred_lgb 注意：以下だとうまく代入できなかった X_train_level2_tmp[dates_train_level2 == cur_block_num][:, 0] = pred_lr X_train_level2_tmp[dates_train_level2 == cur_block_num][:, 1] = pred_lgb			

343	関数を変えてループを回したい	関数を変えてループを回したい	関数を変えてループを回したい	以下の容量で実施可能
				
				# Min value
				f_min = lambda x: x.rolling(window=3, min_periods=1).min()
				# Max value
				f_max = lambda x: x.rolling(window=3, min_periods=1).max()
				# Mean value
				f_mean = lambda x: x.rolling(window=3, min_periods=1).mean()
				# Standard deviation
				f_std = lambda x: x.rolling(window=3, min_periods=1).std()
				
				function_list = [f_min, f_max, f_mean, f_std] # 関数そのものをリストの成分にできる
				function_name = ['min', 'max', 'mean', 'std']
				
				for i in range(len(function_list)):
				  train_monthly[('item_cnt_%s' % function_name[i])] = train_monthly.sort_values('date_block_num').
				groupby(['shop_id', 'item_category_id', 'item_id'])['item_cnt'].apply(function_list[i])
				
				# Fill the empty std features with 0
				train_monthly['item_cnt_std'].fillna(0, inplace=True)
						https://www.kaggle.com/dimitreolive	

344							
345							