

Time Series

时间格式统一

`df['Date'] = [datetime.strptime(x, "%b-%y") for x in df.Period]` 新建一列存放时间，依据原本period列中 Jan-2021的日期形式改成2021-01-01的标准形式

`df_b['Date'] = [datetime.strptime(x, "%d/%m/%y") for x in df_b.iloc[:,0]]`

列名操作

`df.columns = ["Period", "GoldPrice", "BondYield"]` 更改列名

`df.columns = [each.strip() for each in df.columns]` 去除列名前后的空格

数据提取

`df_gold = df[df.Gold.isna() == False].loc[:, ["Top Producers", "Gold"]]` 取指定列的非空行

绘图

`df.groupby("Date").Gold.mean().plot(kind="line")` 按日期分组，统计黄金的均值，并绘制折线图

`df.loc[:, ["Gold", "Silver", "Date"]].groupby("Date").mean().plot()` 按日期分组，同时统计黄金、银的均值，并绘制折线图

`df.groupby("month")[["column0", "column1", "column2"]].mean().plot()` 写法2

`df.plot(kind="line", y="Gold ETF", x="Date")` 指定横纵坐标绘制折线图

`sns.heatmap(df.loc[:, ["Gold", "Platinum", "Silver"]].corr(), annot=True)` 先计算黄金银铂金两两之间的相关关系，并依据结果绘制热力图

`sns.barplot(data=df_platinum, y="Platinum", x="Top Producers")` 运用seaborn绘图 (import seaborn as sns)

Boost Classifier

SVM (cont)

AdaBoost	
from sklearn.ensemble import AdaBoostClassifier	导入AdaBoost
classifier = AdaBoostClassifier(n_estimators=3, learning_rate=0.2, random_state=0)	参数设置
classifier.fit(x_train, y_train)	y_pred = classifier.predict(x_test)
from sklearn.model_selection import GridSearchCV	最优参数选择
param_grid = {'n_estimators': [1, 10, 100], 'learning_rate': [0.2, 0.4, 0.6, 0.8]}	grid = GridSearchCV(AdaBoostClassifier(), param_grid, scoring='accuracy')
grid.fit(x_train, y_train)	grid.best_estimator_, grid.best_params_, grid.cv_results_
GradientBoost	
from sklearn.ensemble import GradientBoostingClassifier	导入GradientBoost
classifier = GradientBoostingClassifier(n_estimators=1, learning_rate=0.4, max_depth=1, random_state=0)	classifier.fit(x_train, y_train)
param_grid = {'n_estimators': [1, 10, 100], 'learning_rate': [0.2, 0.4, 0.6, 0.8]}	grid = GridSearchCV(GradientBoostingClassifier(), param_grid, scoring='accuracy')
grid.fit(x_train, y_train)	grid.best_estimator_, grid.best_params_, grid.cv_results_

SVM	
标准化处理，将数字型的变量转换为-1到1的区间里	from sklearn.preprocessing import StandardScaler
ss = StandardScaler()	x_transformd = ss.fit_transform(x)
分割训练集和测试集	from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_transformd, y, test_size=0.3, random_state=0)	按照训练集70%划分
支持向量机回归	from sklearn.svm import SVR
y_pred_SVR = regression.predict(x_test)	用训练集做拟合，测试集预测结果
regression = SVR()	regression.fit(x_train, y_train)
支持向量机分类器	from sklearn.svm import SVC
df["y"] = pd.cut(x = df.col0, bins=[0, 6, 10], labels=[0, 1])	依据col0列的值划分成两组，0-5一组，6-10另一组，并保存在新列中 (categorical Y适用于分类器)
classifier = SVC(kernel=kernel, random_state=0)	kernel还可选择'linear', 'poly', 'rbf', 'sigmoid'，准确率随之改变
classifier.fit(x_train, y_train)	y_pred = classifier.predict(x_test)
结果评估	
回归适用	from sklearn.metrics import mean_squared_error, mean_absolute_error
mean_absolute_error(y_test, y_pred_SVR)	MAE预测值和实际值之间绝对误差的平均值
mean_squared_error(y_test, y_pred_SVR)	MSE预测值和实际值之间误差的平方的平均值
分类适用	from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
accuracy_score(y_test, y_pred)	accuracy = (TP+TN)/(TP+TN+FP+FN)
confusion_matrix(y_test, y_pred)	详细的TP, TN, FP, FN的数量
classification_report(y_test, y_pred)	包括了准确率、召回率、F1分数和支持度等指标



By **Molly_6075**
cheatography.com/molly-6075/

Not published yet.
 Last updated 16th October, 2023.
 Page 2 of 4.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Feature Engineering

<code>pip install scikit-image</code>	安装包
读取图片并展示	<code>from skimage import io</code>
<code>food = io.imread("chips1.jpg")</code>	<code>io.imshow(food)</code>
变化图片颜色	<code>from skimage.color import *</code>
<code>io.imshow(rgb2gray(food))</code>	将彩色图片转换为灰色并展示
给图片加滤镜	<code>from skimage.filters import *</code>
<code>io.imshow(laplace(food, kernel_size=3, mask=None))</code>	使用Laplace filter并设置内核大小为3
更改图片尺寸	<code>from skimage import transform</code>
<code>image = transform.resize(image, (2000, 2000)) print(image.shape)</code>	更改为指定尺寸并检查更改后的大小
主成分分析	<code>from sklearn.decomposition import PCA</code>
<code>pca = PCA(n_components=30).fit(chip)</code>	降维并保留30个主成分，将PCA模型拟合到chip图像数据上
<code>x_new = pca.transform(chip)</code>	使用已经训练好的PCA模型，将chip图像数据投影到新的特征空间中（包含30个重要特征）
<code>reodata = pca.inverse_transform(x_new)</code>	建了图像数据，只使用了30个主成分来表示原始图像
<code>os.listdir(".")</code>	列出当前工作目录内的所有文件名和目录名
<code>os.chdir("directorypath")</code>	改变工作目录
<code>os.getcwd()</code>	获取当前的工作目录

K-NN

导入K-NN	<code>from sklearn.neighbors import KNeighborsClassifier</code>
<code>classifier = KNeighborsClassifier(n_neighbors=6)</code>	<code>n_neighbors</code> 的默认值是5
<code>classifier2.fit(x_train, y_train)</code>	训练模型
<code>accuracy_score(y_test, classifier2.predict(x_test))</code>	得出准确率

Text Analytics

<code>s.strip</code>	删除尾端全部空格，s为字符串名
<code>a.upper()</code> / <code>a.lower</code>	将字符串a转换成大写、小写形式
分词 Tokenization	<code>import nltk</code>
<code>tokens = nltk.word_tokenize(text)</code>	读取长文本并根据空格和标点分词
打开文件并读取	<code>with open('sample.txt', 'r', encoding='utf-8') as f: tokens = nltk.word_tokenize(f.read())</code>
词性标签 Part-of-speech (POS) Tagging	<code>nltk.download('averaged_perceptron_tagger')</code>
<code>tagged = nltk.pos_tag(tokens)</code>	给分词后的每个单词加个标签
删除前后缀 Stemming (可能产生invalid word)	<code>from nltk.stem import PorterStemmer</code>
<code>ps = PorterStemmer()</code>	<code>print(ps.stem('campaigning'))</code>
词形还原 Lemmatization (generally valid)	<code>from nltk.stem import WordNetLemmatizer</code>
<code>wnl = WordNetLemmatizer()</code>	<code>wnl.lemmatize('beaten', 'v')</code>
情感分析	<code>from nltk.sentiment.vader import SentimentIntensityAnalyzer</code>
<code>nltk.download('vader_lexicon')</code>	<code>analyzer = SentimentIntensityAnalyzer()</code>
<code>analyzer.polarity_scores(text)['compound']</code>	分析长文本的情感色彩并得出综合分数



By **Molly_6075**
cheatography.com/molly-6075/

Not published yet.
 Last updated 16th October, 2023.
 Page 3 of 4.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Text Analytics (cont)

<pre>for index, row in df.iterrows(): compound_score = analyzer.polarity_scores(row['clean_text'])['compound']</pre>	dataframe中按行读取cleaned_data列的每一条数据，并得出综合得分
--	---

Web scrapping

<pre>import request</pre>	用于发送 HTTP 请求
---------------------------	--------------

<pre>response = request.get(url)</pre>	获取数据
--	------

<pre>result = response.json()</pre>	加工数据并print
-------------------------------------	------------

Beautiful Soup

<pre>from bs4 import BeautifulSoup</pre>	解析和处理网页
--	---------

<pre>r = requests.get(url)</pre>	请求网址，其中url为包含网址的变量
----------------------------------	--------------------

<pre>print(soup.title)</pre>	获取网页的标题
------------------------------	---------

<pre>soup = BeautifulSoup(r.content, 'html.parser')</pre>	解析获取到的内容
---	----------

<pre>title = soup.find_all("h6", "h6 list-object__heading")</pre>	运用find_all查找指定内容第一个变量是tag,第二个变量为class (查找新闻标题)
---	--

<pre>each_title = title.text</pre>	通过.text获取标题内容
------------------------------------	---------------

<pre>each_title = each_title.strip()</pre>	删除标题前后的空格，之后print(each_title)
--	-------------------------------

<pre>data2 = r.json()</pre>	将请求的结果转换成json字符串
-----------------------------	------------------

<pre>data2.keys()</pre>	查看json中包含的键
-------------------------	-------------

<pre>data2['help']</pre>	help是其中一个键名(keys)
--------------------------	-------------------

<pre>data2['result']['records']</pre>	直接通过json中的层级关系查找内容
---------------------------------------	--------------------



By **Molly_6075**
cheatography.com/molly-6075/

Not published yet.
Last updated 16th October, 2023.
Page 4 of 4.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>