パターンテンプレート

1.2 LOGISTIC REGRESSION（TFIDFVECTORIZER活用編）

デザインパターン

- パターン名: Logistic Regression（テキスト分析）パターン
- 分類名:「分類」（線形分離可能な教師あり学習）
- 目的:既知の量的変数から未知の質的変数を予測する

■ロジスティック（テキスト分析）回帰が適している例

- スパムメールとそうでないメールを分類する

問題:

- 問題: Logistic regression with words and char n-grams
  - https://www.kaggle.com/tunguz/logistic-regression-with-words-and-char-n-grams
- 内容:
  - Wiki文章をネガティブなキーワードで分類する
- 予測値の評価
  - 正解率：全データのうち、正しく予測（"1", "0" いずれも）できた割合
  - 精度："1"と予測したデータのうち、実際に"1"だった割合
  - 再現率：実際に"1"であるデータのうち、"1"と予測できた割合
  - F値：精度と再現率の調和平均（精度と再現率はトレードオフの関係にあるため、バランスをとった指標として使われる）

適用条件

- 説明変数にテキストデータが含まれること。

適用手順

1. テキストデータを含む訓練/評価データを読み込む
2. データを観察する（図表を使う）
3. 必要に応じデータを整形する（欠損値、カテゴリ属性）
4. テキストデータをTfidfVectorizerを使って数値データに変換する
5. ロジスティック回帰 分析器のハイパラメータを決定する
6. 数値変換後のテキストデータ（訓練/評価データ）を使って、学習モデルを作成する
7. 学習モデルのスコア値（正解率など）を算出する。
8. 学習後モデルに評価データを適用し目的変数の予測を行う

実装上の注意点（scikit-learnを使う前提）

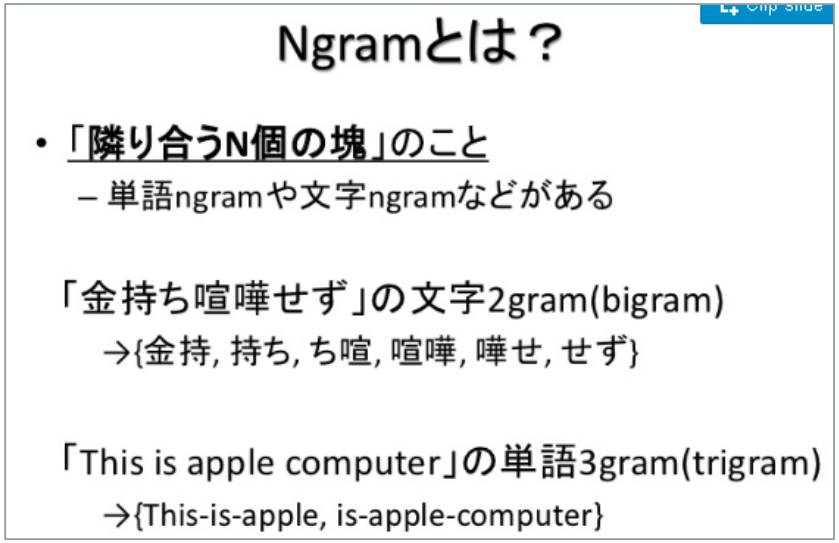
＜ TfidfVectorizerの引数のうち注意が必要なもの＞

■抽出パターン　token_pattern

- デフォルトは2文字以上に単語に設定されているので、1文字の単語を抽出できるように、デフォルト値を変えること。
  - 具体的には「 r'¥w{1,}'」を指定すればよい。

■N-gram範囲　ngram_range

- 一般的に、1～3の範囲内にする
- N-gramについては右図に記載



https://www.slideshare.net/phyllo/ngram-10539181

サンプルコード

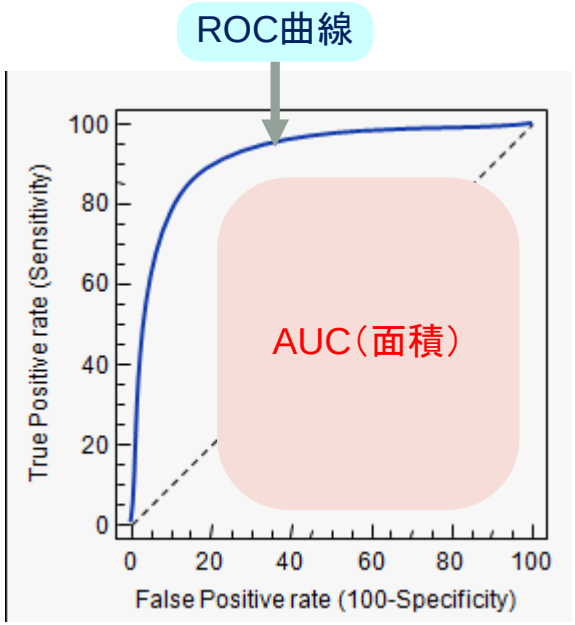
- Github
  - https://github.com/topse2018-kaggle/team/tree/master/oouchi/1.2_LogisticRegression_TextAnalyze

ファイル名	説明
kaggle-tfid-lg.ipynb	サンプルコード（IPython）
kaggle-tfid-lg.html	サンプルコード（結果付き）のHTML出力
train_light.csv	訓練データ（サンプルコードの負担を減らすため、train.csvの一部を切り出したもの）
test_light.csv	評価データ（サンプルコードの負担を減らすため、train.csvの一部を切り出したもの）
submission.csv	Kaggle提出用データ（訓練/評価データをありのまま使っていないため未提出）
1.2_RogisticRegression(TextAnalyze).pptx	本ファイル

適用結果

Score	AUC（※）
toxic	0.9397
severe toxic	0.9774
obscene	0.9677
threat	0.9771
insult	0.9571
identity_hate	0.9516
Total	0.9617

（※）ROC曲線の横軸と縦軸に囲まれた部分の面積。1に近づくほど良いモデル。分類モデルのパフォーマンス評価指標としてよく使われる。



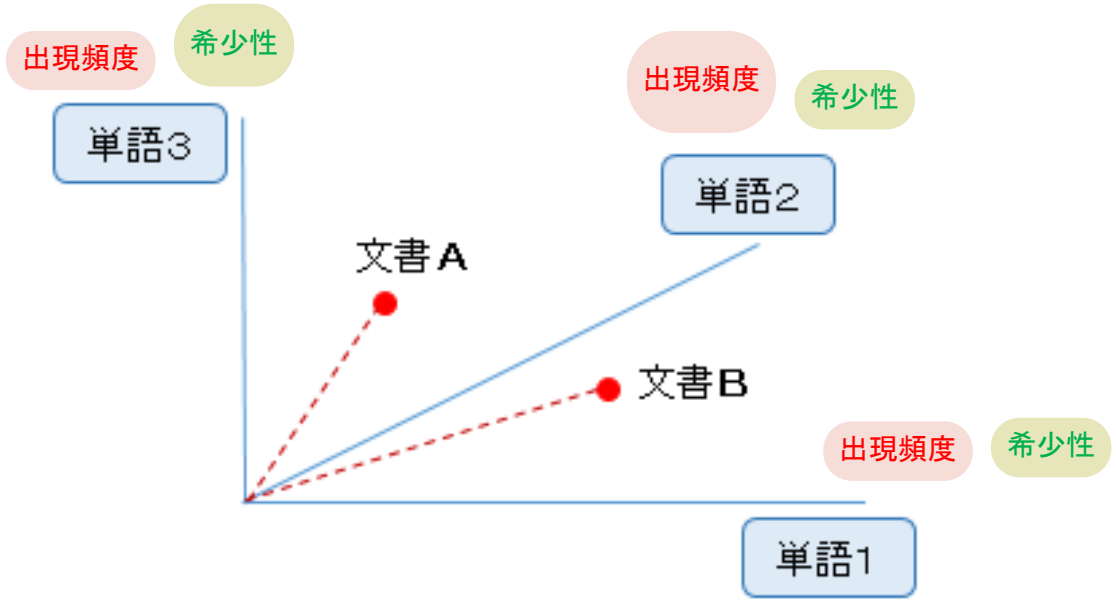
キーワード1つ1つに対して、ロジスティック回帰を適用し、AUCを求めている。よって、多クラス分類（OVR）ではない。詳細はサンプルコードに記載してある。

理論的背景（目次）

1. 考え方
2. TF-IDFとは
3. TF-IDFを求める「TfidfVectorizer」について
4. 「TfidfVectorizer」の主なパラメータ

理論的背景(1. 考え方)

- 文章の特徴を、各単語の出現頻度と希少性を要素とするベクトル(大きさと方向)で表す。
- スパムメールの振り分けをはじめ、様々な用途がある。



理論的背景(2. TF-IDFとは)

- 単語の出現頻度(TF)と希少性(IIDF)を二つ掛け合わせた値
- TF (Term Frequency) : 単語の出現頻度
- IIDF(Inverse Document Frequency) : 希少性

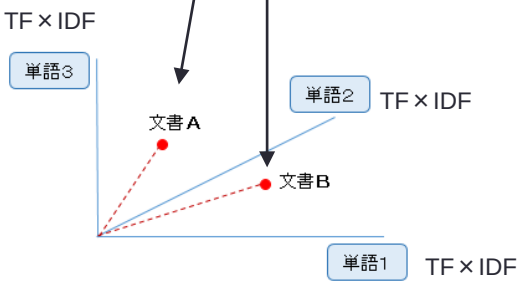
$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

単語の出現頻度

希少性

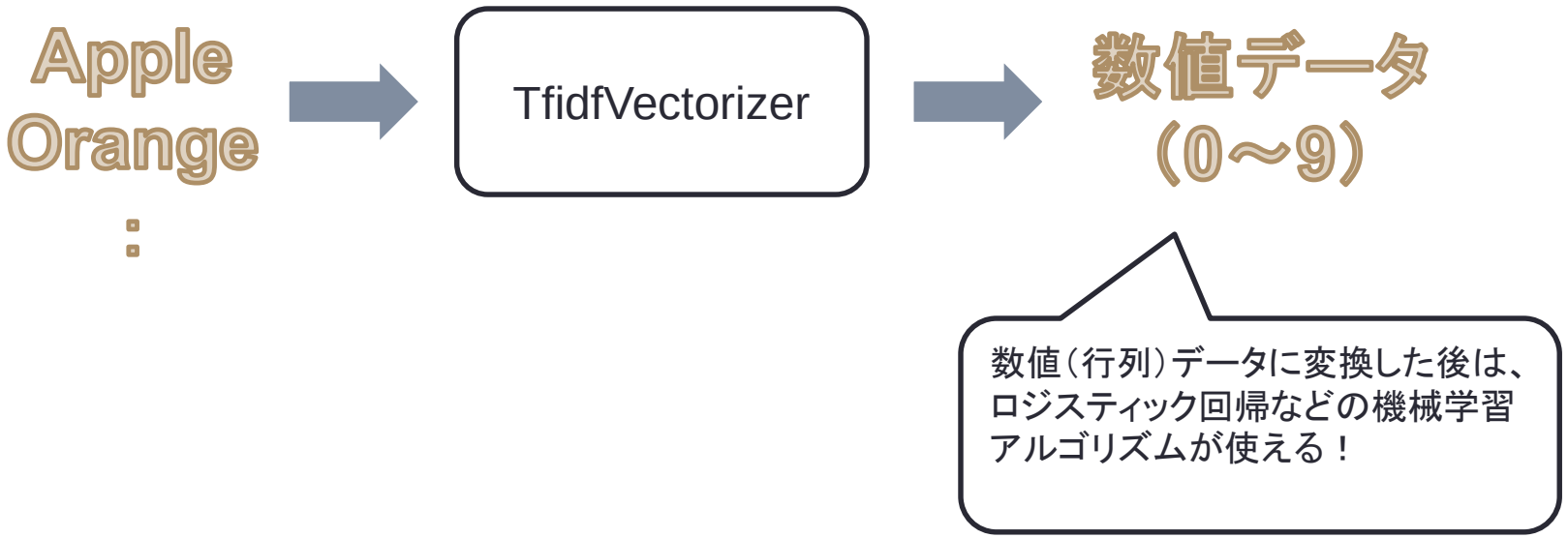
文書 d に含まれる単語 t の数

$$\log \left[\frac{\text{文書の総数}}{\text{単語 t を含む文書数}} \right]$$



理論的背景（3. TF-IDFを求める「TfidfVectorizer」について）

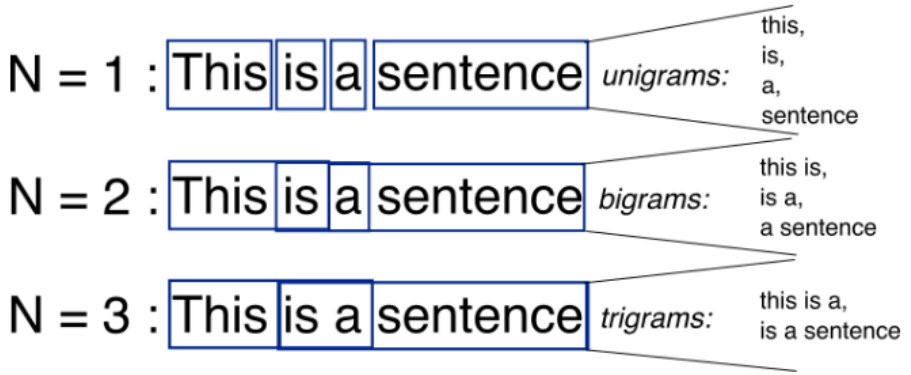
- 「TfidfVectorizer」はTF-IDF値を求めるscikit-learnライブラリ
- テキストデータを数値（IT-IDF値）することによって、ロジスティック回帰をはじめとした様々な機械学習アルゴリズムで扱うことが可能になる



理論的背景（4.「TfidfVectorizer」の主なパラメータ）

パラメータ	説明
sublinear_tf	TrueのときTF値を対数変換する。これにより出現頻度が極端に高い単語に引きずられにくくなる。
strip_accents	アクセント記号（ドイツ語のウムラウトとか）を取り除く場合に使う文字コード。
analyzer	分析単位（単語 or 文字）を指定する。
token_pattern	単語の抽出パターン
stop_words	Tfidf値を求めなたくない単語のリストを指定する。
max_features	抽出する単語数を指定する。
ngram_range	N-gramの上限と下限を指定する。

■例（"This is a sentence"に対する単語N-gram）



出典

- TF-IDF で文書をベクトル化。python の TfidfVectorizer を使ってみる
  - http://ailaby.com/tfidf/
- TfidfVectorizerのよく使いそうなオプションまとめ
  - http://moritamori.hatenablog.com/entry/tfidf_vectorizer
- scikit-learnでtf-idfを計算する
  - https://qiita.com/katryo/items/f86971afcb65ce1e7d40
- Scikit-learn本家
  - https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

関連するパターン

- 線形分離可能な観点
  - 線形回帰　LinearRegression
  - Lasso回帰（L1正則化）　Lasso
  - Lasso回帰（L2正則化）　Ridge
- 「分類」観点
  - 線形サポートベクターマシン　LinearSVC
  - 決定木　DecisionTreeClassifier
  - サポートベクターマシン　SVC
  - ニューラルネットワーク ※　パーセプトロン、CNN、RNNなど
  - K近傍法　KNeighborsClassifier

※ TensorFlow/Keras のみ