OVERVIEW

The Generic Text Plugin (GTP) takes in a dataset column that is assumed to be converted to a list of strings. The GTP then removes any null (None) values found in the list. Then, the GTP returns a score for each value based on how well the GTP is confident that that value is of type generic text. After scores have been assigned, any outlier scores (anything not within 2 standard deviations from the mean) are removed before taking and returning the final average of the scores (the final confidence score for the entire column).
Note: The Generic plugins are used for the Fallback column in the Catalog.

FUNCTION DEFINITIONS

---

get_confidence_score()

The *get_confidence_score()* function takes in a list of *strings* called *col*. *col* is presumed to be a dataset column that has been converted to a list of *strings*. The function then creates an empty *double* list called *scores* to contain the confidence score for each value in *col*. The function then modifies *col* by calling on the *remove_null()* function. After, the function iterates through each value in *col* and calls on the *get_elem_score()* function to append each score returned from that function to the *scores* list. Afterwards, the *remove_outliers()* function is called on the *scores* list. Finally, the average (which is *double* type) of the *scores* list is returned (this is the final confidence score for the entire column).

get_elem_score()

The *get_elem_score()* function takes in a *string* called *elem*. *elem* is presumed to be a value from a list of *strings*. The function then uses regular expressions in an if-statement to determine what confidence score (which is *double* type) to return. *See the REGEX EXPLAINED section for more information on the confidence score.*

remove_null()

The *remove_null()* function takes in a list of *strings* called *col*. *col* is presumed to be a dataset column that has been converted to a list of *strings*. The function uses list comprehension to return a list where any *None* values have been removed.

remove_outliers()

The *remove_outliers()* function takes in a list of *doubles* called *scores*. *scores* is presumed to be the list of confidence scores of each value in a dataset column. The function first creates an empty *set* to contain outliers found. The function then calculates the average and the standard deviation of *scores*. Then the function iterates through each score in *scores* and determines whether the score is within 2 standard deviations from the average. If the score is not found within that range, that score is added to the outlier list. Because 95% of values in any distribution (*Statistic Found Here*) fall within that 2 standard deviation range, generally only extreme outliers will be removed. Finally, list comprehension is used to remove any found outliers from the *scores* list then that modified list is returned.

REGEX EXPLAINED

Regex Form	Confidence Score
[a-zA-Z]	100.0

Generally, there is one regex expression per confidence score category.
If none of the above formats are matched, a confidence score of 0.0 is returned.

Because this is a generic plugin and given its broad type, a single letter detected should suffice for the detection of generic texts.

However, both generic date and generic time can include letters and thus the detection of time zones *[AST, ADT, EST, EDT, CST, CDT, PST, PDT, AKST, AKDT, HST, HDT, UTC]*, night and day distinctions *[AM, PM]*, months [*Jan, Feb, March, April, May, June, July, Aug, Sept, Oct, Nov, Dec*] (*also denoted with all capitalization)* and weekdays *[Mon, Tues, Wed, Thurs, Fri, Sat, Sun]* (*also denoted with all capitalization)* will result in -2.0 deduction in score per date/time indicator found. This deduction deviates from the usual 5.0 deduction because it is very possible that a text can still be classified as generic and have date/time indicators thus a smaller deduction is needed.

These confidence score pairings are subject to change either to team majority vote or mentor discretion.

IMPLICATIONS FOR FUTURE

In the future, we would like to better handle not counting pure dates or times. This could be done with more fortified regular expressions or an external API to detect date and time.