OVERVIEW

The Credit Card Expiration Date Plugin (CCEDP) takes in a column name and dataset column values that are assumed to be converted to a list of strings. The CCEDP first removes any leading and trailing spaces in the strings. The CCEDP then removes any null (None) values or any string values denoting a null value ('NA', etc.) found in the list. Then, the CCEDP returns a score for each value based on how well the CCEDP is confident that that value is a credit card CVV number. 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).

FUNCTION DEFINITIONS

DEPENDENCIES: re

get_confidence_score()

The *get_confidence_score()* function takes in a *string* called *col_name* and a list of *strings* called *col_vals*. *col_vals* is presumed to be a dataset column that has been converted to a list of *strings*. *col_name* is presumed to be the dataset column name. The function first checks to see if '*expiration'* is detected within the column name. If so, the *expiration_name_check* is set to *True* and *False* otherwise. The function then checks to see if '*credit'* or *'card"* is detected within the column name. If so, the *credit_card_name_check* is set to *True* and *False* otherwise. The function then creates an empty *double* list called *scores* to contain the confidence score for each value in *col_vals*. The function then modifies *col_vals* by calling on the *remove_lead_trail_space()* then the *remove_null()* function. After, the function iterates through each value in *col_vals* 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 and is capped at 100.0) or 0.0 is returned if *scores* is empty.

get_elem_score()

The *get_elem_score()* function takes in two bools called *expiration_name_check* and *credit_card_name_check* and a *string* called *elem*. *expiration_name_check* is presumed to be *True* if '*expiration'* is detected in the column name and *False* otherwise. *credit_card_name_check* is presumed to be *True* if '*credit' or 'card'* is detected in the column name and *False* otherwise. *elem* is presumed to be a value from a list of *strings*. Depending on which *name_check* variables are true, a certain score multiplier is set.The function then uses regular expressions in each if-statement to determine what confidence score (which is *double* type) to return. For whatever non-zero confidence score is chosen, a score multiplier will be applied. *See the REGEX section for more information on the confidence score and score multipliers.*

remove_null()

The *remove_null()* function takes in a list of *strings* called *col_vals*. *col_vals* 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 and where any strings denoting null values have been removed. The list of strings that denote values are noted as: *['NA', 'N/A', 'na', 'n/a', 'Na', 'N/a', '']*.

remove_lead_trail_space()

The *remove_lead_trail_space()* function takes in a list of *strings* called *col_vals*. *col_vals* 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 all the string elements have leading and trailing space 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. *scores* is immediately returned if the *scores* list is empty. 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

Conversion Per Format

Format	Regex Form	Confidence Score
09/23	(0[1-9]/\d{2})	100.00
12/20	(1[0-2]/\d{2})	100.00
9/23	[1-9]/\d{2}	70.00
AUG 2023	[JAN|FEB|MARCH|APRIL|MAY|JUNE|JULY|AUG|SEPT|OCT|NOV|DEC|Jan|Feb|March|April|May|June|July|Aug|Sept|Oct|Nov|Dec][a-zA-z]*\s\d{4}	60.00
Aug 2023	[JAN|FEB|MARCH|APRIL|MAY|JUNE|JULY|AUG|SEPT|OCT|NOV|DEC|Jan|Feb|March|April|May|June|July|Aug|Sept|Oct|Nov|Dec][a-zA-z]*\s\d{4}	60.00
August 2023	[JAN|FEB|MARCH|APRIL|MAY|JUNE|JULY|AUG|SEPT|OCT|NOV|DEC|Jan|Feb|March|April|May|June|July|Aug|Sept|Oct|Nov|Dec][a-zA-z]*\s\d{4}	60.00

Conversions Per Confidence Score

Note: These regex forms are used in the if-else statements.

Regex Form	Confidence Score
(0[1-9]/\d{2})|(1[0-2]/\d{2})	100.0
[1-9]/\d{2}	70.0
[JAN|FEB|MARCH|APRIL|MAY|JUNE|JULY|AUG|SEPT|OCT|NOV|DEC|Jan|Feb|March|April|May|June|July|Aug|Sept|Oct|Nov|Dec][a-zA-z]*\s\d{4}	60.0

Name Checks and Score Multipliers

Note: These name checks are used in the if-else statements.

Name Checks Found	Score Multiplier
expiration and (*credit* or *card*)	x1.1
credit or *card*	x0.8

expiration	x0.6
None	x0.5

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.

This regular expression is used with the *re.fullmatch()* function therefore any additional or missing items from the regular expression will cause a score of 0.0.

Credit card expiration dates are usually denoted with a 2-digit month format and 2-digit year format. For the 2-month format in the above regular expressions, *00* and any two-digit number above *12* is not accepted [since those numbers do not denote months].

Because the expiration date can signify other dates, the column name is highly important to distinguish between a credit card expiration date and a non-credit-card-expiration date.

If *'expiration' and 'credit' or 'card'* is found in the column name then a score multiplier of x1.1 is set. If only *'credit' or 'card'* are found in the column name then a score multiplier of x0.8 is set *(this reduction is set as it could mean when the card was acquired, etc.)*. If only '*expiration*' was found then a score multiplier of x0.6 is set *(this reduction is set as it could be an expiration date for another thing)*. If none are found, then a score multiplier of x0.5 is set *(this reduction is set as this could now just be a generic date).*

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

IMPLICATIONS FOR FUTURE

Since credit card information is highly sensitive information, we would like to add a flag that denotes this specific data type (credit card expiration date) as sensitive.

Since this plugin uses the same format as other plugins that use regular expression, we would like to have one .txt file containing all data types' regular expressions and one Python file that handles all regular expression cases instead of having separate Python files for each data type.