City Plugin Documentation

Overview

The City Plugin helps preprocess and analyze a dataset column to calculate a confidence score for how likely each value in the column is a valid US City. The plugin uses a column name and a database of over 108k US cities and small towns to evaluate the results.

Dependencies/ Pre-requisites

Regular Expression module - **Re**
statistics module
state_city_database.py / us_cities- a Python file that contains a data frame us_cities containing all the US cities and small towns.
fuzzywuzzy library- A library that checks the similarity between two strings (to catch spelling errors) To install:

1) Open the command prompt/ terminal window
2) Type **"pip install fuzzywuzzy"**
3) Type **"pip install python-Levenshtein"**

Fuzzywuzzy library has a dependency on the Levenshtein module, install the module to avoid any warnings/errors.

Function Definitions

get_confidence_score()
The get_confidence_score() function takes in the column name as a string and a sample of the dataset column as a list of strings.

The function first checks if the column name contains the substring "city" or "cities". A boolean variable "col_check" acts as a checker set to True if the column name matches the condition.

The next step is to preprocess the data: the list of values is sent to functions that remove all the null values and leading/trailing spaces from the data.

The function then initializes a list of scores to keep track of all the confidence score values for the column. Each value is checked through the valid_city() function and the scores list is appended accordingly. Finally, a function removes any outliers in the results and returns a median value which is the final confidence score of the entire column.
Variables used:
col_name- column name (String)
col_values- column values (list of Strings)
col_check- column name checker (Boolean)
scores- confidence score for each value (list of Integers)
valid_city_check- valid state checker (Boolean)
Input- col_name, col_values
Output- median value of scores (float)

valid_city() /state_city_database.py
This function checks each value of the column through the state_city_database.
This database contains a data frame cidties_df) of all US cities which is imported from the us_cities.csv file
Note: All the city values are converted to lowercase and stored in a set to avoid duplicate values.
Therefore the valid_city() function converts the passed values into lowercase and checks them through the set.

1) The function checks if the values are in the database or even a close match.
   The fuzzy-wuzzy library is added to check if the value is a close match in case of any spelling errors. It measures the similarities of two strings using several algorithms. In this code, it is checked if the strings are at least a 70% match

   The any() function is used to return True as soon as there is a matching element, it improves performance by avoiding unnecessary iterations.

2) Additionally, if the value or its close match is not found in the database, it also checks through various common city suffixes like beach, ville, port (long beach, Asheville, gulfport), etc.

Variables used:
city_name: state name/value to be checked through the function (String)
us_cities: A data frame imported from the state_city_database.py / us_cities.csv
Input- state name/ value to be checked
Output- True/False (Boolean)

remove_null()
This function removes all the null and missing values from the data. Values like NA, null, and Nan are removed to avoid skewness in the results.

remove_spaces()
This function removes all leading and trailing spaces (if any) from the values. This ensures that the data is properly formatted and consistent, making it easier to work with and reducing the chances of errors in the code.

remove_outliers():
This function takes in the final scores/confidence values. 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 value 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.

Conditional Reasoning:

Based on the final values of the checkers: colname_check and valid_state_check, the get_sconfidence_score() assigns the following confidence scores:

Condition	Score	Reasoning
col_ckeck - True Valid_state_check - True	100	If the column name matches and the value is found in the database, it is certain that the value is a valid US City and is given a 100% score.
col_ckeck - True Valid_state_check - False	40	If the column name matches but the value is not found in the database, it could be possible that there is an undetected spelling error/missing value. Since the value is checked through an extensive database it is given a higher priority hence the score is still 40% to pass the column name parameter.
col_ckeck - False Valid_state_check - True	60	If the column name does not match but the value is found in the database, the score is set to 60% since it has a higher priority/ more chances of the value being a city.
col_ckeck - False Valid_state_check - False	0	If neither of the conditions is satisfied, it is certain that the value is not a valid US city and the score is set to 0%

Example:

Input:

col_name= “City”

col_values= [“Los Angeles, California”, “neyork”, “Seattle, WA”, None, “n/a”, “93403”, “miami”]

After these values are passed through the plugin, these values will be set as scores

scores=[100.0, 100.0, 100.0, 40.0, 100.0]

Note that the None and n/a values were removed by the remove_null() function, neyork is still given a 100 score because it is a spelling error and gets caught by the fuzzywuzzy library.

The outliers are further removed by the remove_outliers() function and the final score of 100.0 is returned after calculating the median of all the scores.

Implications for the Future:

In the future, we would like to handle international cities. This means adding more databases of cities all around the world or using an external API to check if a value is a valid City or not.