2021

Address Linkage Key

KSU CAPSTONE PROJECT SPRING 2021

KATHRYN GREER, SARITHA GUDALA, KATHLEEN MCELVEEN, BYRON SMITH

PROJECT SITE:
https://sarithavikram.github.io/CapstoneProject_AddressLinkageKey/projectdocs.html

Executive Summary

AnalyticsIQ, the project sponsor, is requesting a tool or set of tools that can be used to calculate the delivery point code (DPBC) of a given address. Having a tool to make this calculation will reduce reliance on third party tools.

The tool(s) will be developed using Python and Spark with version control through Git.

This will be an iterative project with the Minimally Viable Product (MVP) starting as a tool that will calculate the DPBC of a properly formatted address using the main USPS rule. Once the first iteration is complete and tested, additional iterations will incorporate further USPS rules and the ability to determine the DPBC of improperly formatted addresses. An ultimate goal for the tool(s) is to determine if specific types of address, such as high-rise buildings, are included in a given dataset.

Project Presentation

Link for Project Formal Presentation:

Link for Project Short Presentation:

Introduction

Business and Project Background

AnalyticsIQ, the project sponsor, is a Marketing Data and Analytics company that works with big data from publicly available sources to allow their clients to deliver personalized experiences to consumers.

Address information is very useful in marketing, and there are existing software options that use the USPS guidelines to calculate information from addresses. AnalyticsIQ would like a tool or set of tools that would accomplish this task, determining the DPBC of a given address, so that address information can be used to help gain insights on the data they are analyzing and providing for clients. Creating a tool(s) would not only reduce reliance on third party software, but also allow for customization such as determining if certain address types, such as a high-rise building, are present in a given dataset.

Project Scope, Objectives, and Deliverables

This project was created to create a tool or set of tools that will determine the DPBC of a given address following the published USPS guidelines for determining DPBCs.

This is an iterative project that seeks to continually refine a minimally viable product (MVP). The first iteration will be to apply USPS rule number 1, which will determine the DPBC for most properly formatted addresses. Once the initial MVP is completed, additional iterations will compute the DPBC using additional USPS rules to process improperly formatted addresses or addresses that contain a secondary address component. Subsequent iterations will work to determine characteristics of addresses in a given dataset, such as high-rise buildings.

Analysis and Design

Technical Background

The tool will be designed using Python and Spark (PySpark) to allow for scalability to use the tool on Big Data datasets.

Version control and task assignments will be handled via Git. AnalyticsIQ created a GitLab site for this project where the code is stored. Task assignments are handled via a Kanban board on the project GitLab site. SSH is used to provide security when downloading or uploading code from the GitLab site to a local machine to work on the code.

Testing and prototyping will be done in Jupyter Notebook. AnalyticsIQ has provided a JupyterLab server instance for the team, and Docker containers will also be used to initialize PySpark-enabled Jupyter Notebooks locally.

Tool Design

The initial design of the tool meets the guidelines set forth by the USPS for determining the delivery point barcode (dpc) of an address using the primary address line 1 field. To accomplish this, a function was added to the tool to accommodate each of the USPS primary address rules as follows.

0. Housenumber: this function creates a column to hold the housenumber, that starts with a digit, of an address from the address_line_1 field.

```
def house_number_extract(df):

   df = df.withColumn('housenumber', (f.regexp_extract(f.col('address_line_1'),'(^[0-9]([0-9A-Z.*-]+)?)', 1)))

   return df
```

1. General Rule: This is the primary USPS rule which creates a dpc for an address where the house number is only digits. For this rule, the dpc is the last 2 digits of the housenumber.

```
def apply_rule_1(df):

   # Use last two digits. Print code characters in dpc representing last two digits of primary street number

   df = df.withColumn('dpc',

      f.when(

         f.col('address_line_2').isNull() &

         f.col('housenumber').isNotNull() &

         f.col('housenumber').rlike('^[0-9]*$'),

         f.col('housenumber').substr(-2,2)))

   return df
```

2. No Numbers: This rule adds the value 99 to the dpc field if the address_line_1 field has no house number. For this tool, this rule is combined with 7, 9, and 13 as they all set the dpc to 99 if specific conditions are met.

```
def apply_rule_2(df):

   # Use 99 when address contains no house number

   df = df.withColumn('dpc',f.when((f.col('dpc').isNull()) & (f.col('housenumber').isNull()),f.lit('99'))

                  .otherwise(f.col('dpc')))

   return df
```

3. Single Digits: This rule ensures a two digit dpc is calculated for addresses where the housenumber is a single digit by adding a leading zero to that value. This function is design to be applied last, in order to correct any single digit dpc values.

```
def apply_rule_3(df):

   # Add a leading 0 when address contains a single digit house number

   # left pads dpc column with 0 up to length 2

   df = df.withColumn('dpc', f.lpad('dpc', 2, '0'))

   return df
```

4. Fractional Number: This rule determines the dpc for an address with a fraction present as part of the address_line_1 value. This rule was determined to be out of scope for this project by the project sponsor, AIQ.
5. Trailing Alphas: This rule determines the dpc for an address where the house number consists of digits and alpha characters where the alpha characters are at the end of the house number.

```
def apply_rule_5(df):

   #Ignoring trailing alphas. Print code characters in DPBC representing last two digits to left of space
and alphas

   df = df.withColumn('dpc',

            f.when(

                f.col('address_line_2').isNull() &

                f.col('housenumber').isNotNull() &

                f.col('dpc').isNull() &

                #f.col('housenumber').rlike('^[a-zA-Z0-9]*$'),

                f.col('housenumber').rlike('^[0-9]+[A-Z]+$'),

                f.regexp_extract(f.col('housenumber'),'(\d+)',1).substr(-2,2)).otherwise(f.col('dpc')))

   return df
```

6. Spaces and Alphas: This rule determines the dpc for an address where the housenumber is numeric and is followed by a space and alpha characters before the street name. This rule does not have a specific function in the tool as it applies the same logic as rule number 1.
7. Alphas Only: This rule adds the value 99 to the dpc field if the address_line_1 field has only alpha characters preceding the street name. For this tool, this rule is combined with 2, 9, and 13 as they all set the dpc to 99 if specific conditions are met.
8. Leading and Embedded Alphas: This rule calculates the dpc when the housenumber portion of the address_line_1 field contains only alpha and numeric characters and contains a leading or embedded alpha character.

```
def blanks_to_null(x):

    return f.when(f.col(x) != "", f.col(x)).otherwise(None)

def apply_rule_8(df):

    # run blank function on 'dpc' column to replace 00s with 'None'.

    df = df.withColumn('dpc', blanks_to_null('dpc'))


    # create new column that selects the first word from the address_line_1 string

    df = df.withColumn('alphas', (f.regexp_extract(f.col('address_line_1'),'(^[A-Z0-9]+[0-9]\\w)', 1)))


    # update dpc when alphas contains a value, add_line_2 is null and dpc is null

    df = df.withColumn('dpc',

        f.when(

            f.col('alphas').isNotNull() &

            f.col('address_line_2').isNull() &

            f.col('dpc').isNull(),

            f.regexp_extract(f.col('alphas'),'([0-9]{1,2}$)',1)).otherwise(f.col('dpc')))

    return df
```

9. Slashes: This rule adds the value 99 to the dpc field if the house number portion of the address_line_1 field contains one or more slashes. For this tool, this rule is combined with 2, 7, and 13 as they all set the dpc to 99 if specific conditions are met.
10. Other Embedded Symbols: This rule calculates the dpc when the housenumber portion of the address_line_1 field contains symbols. For this tool the symbols *,- were assumed.

```
def apply_rule_10(df):

    #Other embedded Symbols: use last 2 digits to the right of the all symbols

    #create a column for the housenumber with the digits after the symbol extracted, this column will be
called symbolHouseNo
```

```
    df1 = df.withColumn('symbolHouseNo', (f.regexp_extract(f.col('housenumber'), '([^.*-]*$)', 1)))

    df = df1.withColumn('dpc',

            f.when(

                # This rule should only run when address_line_2 is null as those addresses will be subject to
different rules.

                f.col('address_line_2').isNull() &

                #This rule should only run when the created column symbolHouseNo is Not Null as that
indcates the presence of a symbol and the address is subject to rule 10.

                f.col('symbolHouseNo').isNotNull() &

                #This specifies that if 'dpc' is not null, then that value should be retained.

                f.col('dpc').isNotNull(),

                f.col('dpc')).otherwise(

                        #If all above criteria are met, the dpc should be the last 2 digits of the the symbolHouseNo.
This does return 1 or 2 digit dpcs, which should be taken care of by Rule 3 later.

                         f.col('symbolHouseNo').substr(-2,2)))

    return df
```

11. Embedded Spaces: This rule determines the dpc for an address with embedded spaces in the housenumber portion of the address_line_1 value. This rule was determined to be out of scope for this project by the project sponsor, AIQ.
12. Numeric Street Names: This rule determines the dpc for an address where the street name is numeric. This rule does not have a specific function in the tool as it applies the same logic as rule number 1.
13. All Other Anomalies: This rule adds the value 99 to the dpc field if none of the other primary address rules apply. For this tool, this rule is combined with 2, 7, and 9 as they all set the dpc to 99 if specific conditions are met.

Implementation

Summary of Tool Design

The tool will be designed using Python and Spark (PySpark) to allow for scalability to use the tool on Big Data datasets.

Version control and task assignments will be handled via Git. AnalyticsIQ created a GitLab site for this project where the code is stored. Task assignments are handled via a Kanban board on the project GitLab site. SSH is used to provide security when downloading or uploading code from the GitLab site to a local machine to work on the code.

Major Features of the Tool

The tool uses the published USPS guidelines to determine the DPBC of a given address.

The first iteration of the tool will be able to calculate the dpc of an address, specifically from the address_line_1 field, using the primary USPS rules.

Subsequent iterations will be able to calculate the dpc of an address with secondary or address_line_2 information.

Final iterations of the tool will work to create data models and gather information about an address based on the address itself, such as whether the address is a high rise.

Conclusion

Project Summary

Reflection on the project, deliverables met, lessons learned, etc.

Limitations and Future Direction

Future optimizations that would increase the usefulness of the tool(s)

References

CASS™ technical guide for CYCLE N. (n.d.). Retrieved February 28, 2021, from https://postalpro.usps.com/CASS/CASSTECH_N

USPS Publication 28. (2020, June). Retrieved 2021, from https://pe.usps.com/cpim/ftp/pubs/Pub28/pub28.pdf

Appendix

Attached Documentation/Information