Data Wrangling Report

1 TABLE OF CONTENTS

2 INTRODUCTION

2.1 BACKGROUND

Data in the real world and that provided for academia are completely different.

Data provided for academia is for teaching purposes and is generally ideal with minimal corrections required to provide examples.

Real world data is messy and untidy with no real structure, and would otherwise overwhelm students.

This projects intent is to simulate in between where real world data has been structured to suit the provider's needs, gathering to put to practice that which has been learnt academically.

2.2 PURPOSE

The purpose of this project is to practically apply the data wrangling process onto a real world application.

The scope of the project involves accessing the following files:

- Archived CSV file, readily provided

- Raw data, scraped via Twitter API

- TSV file, to be downloaded via URL link provided

The above files are to be imported into a Jupyter notebook and prepared for initial analysis.

Then proceeding onto assessing the raw data against end goal objectives and prioritising cleaning steps required to achieve the analysis in question.

Generally, the end result will be a clean set of data, to be written to file and then analysed against the initial data imported prior to the data wrangling process in order to quantify and signify the importance of cleaning.

In order to achieve this several iterations are required factoring time and project constraints to assist in whether cleaning is sufficient.

3 METHODOLOGY

The main four processes in data wrangling consist of:

Gathering – Data collection of structured and semi-structured data types and ingesting it into a data frame, specifically:

- Client provided files in the form of CSV, TSV, txt, JSON, etc.
- Extracting from a database and/or link, pre-formatted based on their structuring methods.
- Scrape from the web providing raw format or,
- Scrape via an API providing semi-structured data.

Transform extracted data from raw file formats, and load into a structured Pandas data frame

Assessing – Reviewing data frames, obtain holistic view of data available and determine questions and or interesting variables to present. Itemize number of quality and structural issues.

Cleaning – Produce code to amend highlighted issues in assessment.

Repeat – Re-iterate as required to achieved above data requirements required for Exploratory Data Analysis (EDA)

3.1 GATHERING

3.1.1 General

General workflow is to write code that is scalable and can be reused for future projects. With this in mind, we will aim to produce code if time permits, with functions for repeated code and other external scripts as required if performing a specific function.

A function is used to create a list of files, this is to allow for future situations where large quantities of files are imported and allows for easier identification if indexed data frames are used, e.g. ***df_raw* [0]** represents the first file imported, and so on.

Importing into the data frame loops operates via a **for loop** scanning the **files list** and applying the appropriate read method into pandas data frame.

Note: Initial thoughts were to provide similar to that of a dictionary whereby accessing the name (key) will provide the data frames name, however has not been implemented and is recorded as part of the future works/function.

Generally opening files use the **with statement** to allow automatic close of file after use to prevent memory issues.

3.1.2 Iteration 1 – Import CSV, download, save and import TSV

First step is anticipating the required libraries for the project and add additional libraries as required along the data wrangling process.

As described in the introduction, import the above files into a panda's data frame for initial analysis. The TSV file requires additional coding as it involves OS manipulation to read and write to project folder and downloading file from provided URL via required library.

3.1.3 Iteration 2 – Scrape from Twitter API, save to file and import

Research and time was required to in order to access the Twitter API. The method to access the API was consolidated into a script that can be utilised for future generalised scraping applications.

User input is covered by a standard **while statement**, **if statements** are utilised as there are no **switch case statements** available in python.

3.2 ASSESSING

3.2.1 General

Assessing involves two assessment methods, to provide a holistic overview and to achieve scalability requirements for large data sets.

Programmatic assessment is the first method of EDA to provide a quick summary of each respective data frame.

Visual assessment, to relate the information summary provided by the programmatic approach. Visually scanning the data and sifting through each columns with context, will allow us to pick up discrepancies and see relationships between columns, if any.

An **assess function** is created to allow to reduce writing repetitive code and store it in a variable for quick access. This function requires additional modification in order to improve readability and accessing i.e. print data frame columns and respective column index.

3.2.2 Iteration 1 – CSV and TSV

An initial visual assessment using **.sample(x)** and programmatic assessment is performed using **.info()**, **.describe()** across all 3 data frames to determine scope of cleaning involved.

Afterwards we breakdown the tasks involved and produce a general checklist, i.e. items that fall under data quality and data structure. Each item is ticked off as each column is scanned to ensure a thorough and consistent assessment.

value_counts() is useful in provide a summary of the value distribution, for binning if required, and finding duplicates. Outliers can be detected provided the results are within the Jupyter notebook display range.

Extracting from HTML tags can be achieved using regex however beautiful soup provides an in-built function that is widely used, whereas regex would involve testing the text against the pattern, adjust as required and apply additional string manipulation depending on catching group.

3.2.3 Iteration 2 – Twitter API

The Twitter API text file is imported into a JSON object that is formatted with the appropriate spacings for readability.

The keys are reviewed and each new entry is extracted out using a **for loop**.

3.3 CLEANING

3.3.1 General

The cleaning stage involved unreferenced copies of the existing data frames to allow cleaning mistakes to be overwritten with the raw values.

Generally quality issues are handled prior to handling structure issues, as joining data frames is not possible without consistent formatting.

The assessment stage and optimizing code exceeded the expected time limit resulting in the cleaning stage to be less presentable.

3.3.2 Iteration 1

3.3.2.1 Quality Issues

1. col0: tweet_id data type change to string, all dataframes

df_twitter

2. col3: change timestamp datatype to datetime

3.1 col4: split string to remove html tag and extract text within

3.2 col4: rename column heading from source to add source_app

4. col1,2,6,7: change datatype from float to string

5.1 remove whitespaces in string/object columns

twitter_image_predictor

5.2 remove white spaces in string/object columns

6. col3,6,9: change to lower case

7. col1: rename from jpg_url to img_url

8. col2: rename from img_num to conf_tweet_img

3.3.2.2 Structure Issues:

1. timestamp split into three columns, date, time, timezone

2. categorize dog type into one column

3. merge, denormalize data frame to contain the relevant columns required for analysis

3.1 twitter_data to contain all relevant twitter data

3.3.3 Iteration 2

3.3.3.1 Quality Issues

9. col12: review col12 to ensure the validity of the names

10. check numerator rating against text and valid/correct

11. remove retweets, indicated by RT @ in text column, **retweet status id** and **in reply to id**

3.4 ITERATING

We iterate through the assessment again programmatically, with our newfound knowledge to dig deeper. And repeat until the data is usable for preliminary analysis because in real life there are deadlines and so practically, we will not be able to assess and clean every fault in the data.

4 CONCLUSION

The data wrangling process can be quite time intensive, as I have discovered throughout this project. Significant amounts of time were invested into learning REGEX, specifically being able to extract the correct names out of the **text column**.

Note: I have excluded this out of the preliminary submission and will continue to attempt to incorporate as part of future commits.

Performing the steps required to assess and clean the data manually has provided me with confidence to tackle future projects, developing a scope of works, creating a trail for others to easily follow and adjust in the future to upgrade and/or scale up/out.

Additional methods of data exporting will be looked into and is part of the project to-do list as showed on the GitHub project plan.