What automated processes can be used?

|  |  |
| --- | --- |
| Manual task performed | Automation (Python) |
| Replace or remove rows with missing values (remove missing ATC) | fillna or dropna |
| Ensure data is within valid range  (ensure all winning prices are below maximum price) | df.drop(df[df['winner\_price'] >= df['maximum\_price\_allowed']].index) |
| Dummy categorical variables | pd.get\_dummies(df['region']) |
| Convert date to month or year | pd.to\_datetime(df['published\_date']).dt.month |
| Duration between two dates | (pd.to\_datetime(df['start\_date']) - pd.to\_datetime(df['published\_date'])).dt.days |
| Add columns together | df['duration']+df['duration\_extension'] |
| Is a column 0 or a value (was the contract extended) | df['duration\_extension'] > 0 |
| Split strings | df['participants'].str.split('|', expand=True) |
| Find difference between winner and second / third/ fourth place | df['winner\_price'] - df[‘second\_place'] |
| Construct logic to find the true winner | np.where((df['outcome'] == 'won') & ((df['second\_place\_outcome'] == 'lost') | df['second\_place\_outcome'].isnull()), df['winner\_price'], df['second\_place\_price']) |
| Conduct z scoring to remove outliers (over 3 sd) | df[columns\_to\_standardize].apply(zscore) |
| Min max normalisation | MinMaxScaler().fit\_transform(df[columns\_to\_normalize]) |

Improvements to long term use

Any data road map needs to have as few points of failure as possible to allow for a smooth flow of raw data to curated result.

1. Documentation should be completed to describe all the steps that were done to achieve the result. This should include instructions for the environment and any naming conventions. Any dependencies which were used need documentation with package names and versions.
2. Identify any manual processes such as emailing CSV files and replace them with automation. As has been done above, any manual transformations should be replaced with processes to eliminate human error.
3. Where humans cannot be replaced, further eliminate human error by applying rules to inputs such as lists and input conditions.
4. Once clear instructions have been completed a foldering system should be created with version controls to maintain order.
5. Schedule work flows to extract data, clean and transform as required. Ensure parallel and sequential processes are identified to avoid race conditions but reduce processing time.
6. Monitor data quality issues such as data types, ranges and duplicates and flag or eliminate these rapidly. Introduce error handling steps to deal with faults in cleaning or transformations.
7. Ensure the process is scalable using load testing experiments and innovate with the suggestions for high volumes below.
8. Test and innovate while maintaining rigour by applying a data science approach of iteratively changing variables, observing the result against known outcomes and logging the results and changes to the base code.
9. Monitor key metrics such as memory use, inconsistencies and failures in order to guide areas for intervention and innovation.

Scale up for high volumes

1. Small scale ups would require processes completed in excel to be conducted in database or in python to allow for higher numbers of rows.
2. Even higher volumes of data may require certain processes to be batched as large data volumes can lead to memory exhaustion.
3. Aggregation functions such as the z scoring used above for anomaly detection, could be replaced with rules based processes, which are validated on samples.
4. Cloud computing may be considered to expand the processing capacity and allow for further scaling without further changes to the process.
5. To reduce processing time the modelling could be simplified by limiting the number of variables striking a trade off between model complexity and accuracy.