Assignment - Creating a model/ an algorithm to order a set of transactions by likelihood of matching a receipt image.

Report by - Ritu Agrawal

Abstract - The assignment tries to help customers map their receipt images with correct transaction. As Tide uses a third party tool that provides the extracted data from the receipts, using that output and creating a sorted dataframe with the correctly mapped/ successfully matched transaction at the top of the dataframe.

Code - Code has been written in Python3 using the Jupyter notebook in Anaconda environment.
The initial cells in the jupyter notebook(Assignment tide) consists of step by step analysis and coding to get the output and the final cell has a module which consists of all the steps which serves as the complete model or algorithm that takes an input as the file location and returns a dataframe with the necessary output.

The logic is: For a transaction to be a successful match, the feature_transaction_id should be equal to the matched_transaction_id and receipt_id should match with the receipt id in the matched_transaction_id. Groupby receipt and transaction id and sort the values by the Success column as Success is 1 when matched else it is 0.

Model/ Algorithm:

```
def preprocess(filelocation):
    """
    Input - filelocation, file type should be a csv
    Output - A dataframe with the correct transaction if exists at the top

    """
    # Converting the csv file to pandas dataframe
    df = pd.read_csv('data_interview_test.csv')

    # First splitting the column names on ':'
```

```
    for i in df.columns:
        column_names = i.split(':')

    # Splitting the values in rows on ':'

    df['list_col'] = df[df.columns[0]].str.split(':')

    # Resetting the index to convert it from multiindex for easier accessibility

    df = df.reset_index()

    # the first three columns should be renamed by the first three names in the column_names
list
    # Splitting the list_col into new columns

    df[column_names[3:14]] = pd.DataFrame(df.list_col.tolist(), index= df.index)

    # Renaming the first three columns

    df.rename(columns={'level_0':'receipt_id', 'level_1':'company_id',
'level_2':'matched_transaction_id'}, inplace=True)

    # Dropping off the columns which are already processed to create new columns

    df = df[column_names]

    # Splitting the matched transaction id based on the ':'

    df['rc_id_mtch_ftr_id'] = df['matched_transaction_id'].str.split(':')

    # The first value in the list contains the transaction id and the second element contains the
receipt id

    df['matched_receipt_id'] = df['rc_id_mtch_ftr_id'].apply(lambda x: x[1])

    # Since, the matched receipt id is object, converting it to int

    df['matched_receipt_id'] = df['matched_receipt_id'].astype(int)

    # Extracting the transaction_id from the matched_transaction_id column

    df['mtchd_trnx_id'] = df['rc_id_mtch_ftr_id'].apply(lambda x: x[0])
```

```
    # Creating a new column with a success value as 1 when the receipt id and the transaction id match else 0

    df['Success'] = np.where((df['mtchd_trnx_id']==df['feature_transaction_id']) & (df['matched_receipt_id']==df['receipt_id']), 1, 0)

    # Groupby receipt and feature_transaction_id, sort by Success value and then, drop the success column

    df = df.groupby(['receipt_id', 'feature_transaction_id']).apply(lambda x: x.sort_values(by='Success', ascending=False)).drop('Success', axis=1)

    # Dropping the unnecessary columns

    df = df[column_names]

    # Returning a dataframe

    return df
```

The steps involved in getting to the answer:

1. Converting the csv file to pandas dataframe
2. Splitting the column names on ':'
3. Splitting the values in rows on ':'
4. Resetting the index to convert it from multiindex for easier accessibility
5. the first three columns should be renamed by the first three names in the column_names list and then, Splitting the list_col into new columns
6. Renaming the first three columns
7. Dropping off the columns which are already processed to create new columns
8. Splitting the matched transaction id based on the ':'
9. The first value in the list contains the transaction id and the second element contains the receipt id
10. Since, the matched receipt id is object, converting it to int
11. Extracting the transaction_id from the matched_transaction_id column
12. Creating a new column with a success value as 1 when the receipt id and the transaction id match else 0
13. Groupby receipt and feature_transaction_id, sort by Success value and then, drop the success column
14. Dropping the unnecessary columns
15. Returning a dataframe

The algorithm is successful as we can see all the correct transactions for a receipt id and transaction id seem to be at the top of the group(group meaning all the records associated with one receipt and one transaction).