CENTRE FOR GOVERNANCE AND SUSTAINABILITY
治理与永续发展研究所
(Formerly known as CENTRE FOR GOVERNANCE, INSTITUTIONS AND ORGANISATIONS)
Spearheading best practices and ideas for corporate governance and sustainability.

SGTI Machine Learning Application

Testing Report

Preparer	Role
Chern Tat Sheng	Researcher

1. Executive Summary **3**

2. Testing Methodology **3**
2.1 Sample selection 3
2.2 Benchmarking 3
2.3 Testing Procedure 3

3. Issues Detected in Testing **5**
3.1 Company names 5
3.2 Missing questions 6
3.3 Matching Errors 6

4. Additional Metrics **7**

5. Limitations **9**

6. Conclusion **10**

1. Executive Summary

The main focus of this report is to detail the testing process on the responses and the scoring of the Machine Learning application, We will select a sample of 57, out of the 519 companies in the scope of the SGTI scoring in FY21 and reconcile the automatically retrieved responses from the application against the manually encoded responses. We will detail the testing process in Section 2 and highlight the key issues detected with the current iteration of the application as well as provide consolidated metrics that may provide insight as to the reliability of the information. Thereafter, the limitations of the testing will be detailed in Section 5.

2. Testing Methodology

2.1 Sample selection

In designing the testing, the sample of 57 companies were selected randomly, by selecting a fraction of companies from each of the 10 industries. Using a stratified sampling technique, the composition and number of companies in each industry within the sample represents the sample fo all the 519 companies that are in scope for FY21. For example, 1 sample for “Energy” and 5 for “Financials” as there are overall more companies in the “Financials” industry compared to the “Energy” industry in scope.

2.2 Benchmarking

For the purpose of testing the results of the application, we must first create an appropriate basis for reconciliation. As such, we will be utilising the manually encoded versions of the responses as well as the scores obtained from the SGTI database (“SGTI data”). The extraction obtained from the application (“Extracted data”), will be compared against SGTI data and comparing the two would give us insight into:

1) Differences between the companies (eg. naming conventions)
2) Differences between the questions (eg. question numbers)
3) Differences between the responses/scores between the Extracted data and SGTI data.

2.3 Testing Procedure

We utilised Python in order to make the task efficient, and designed a structured algorithm to compare the Extracted data against the SGTI data. The code, appended in a Jupyter notebook, will be attached in the zip file and can be reused to obtain similar metrics on future updates over the application. Additionally, the notebook can also be accessed and viewed using this link to my personal GitHub:

https://github.com/Tat-sheng/sgti_testing/blob/main/SGTI%202021%20-%20Comparison.ipynb

An excel spreadsheet is created with 4 sheets, containing the Extracted data for responses and scores from the application (2 Sheets, 57 sampled companies) as well as the SGTI data for responses and scores that were manually encoded by CGS (2 Sheets, 519 sampled companies). In order for the code to work in subsequent tests, kindly format the 4 sheets in a similar manner to the attached excel spreadsheet.

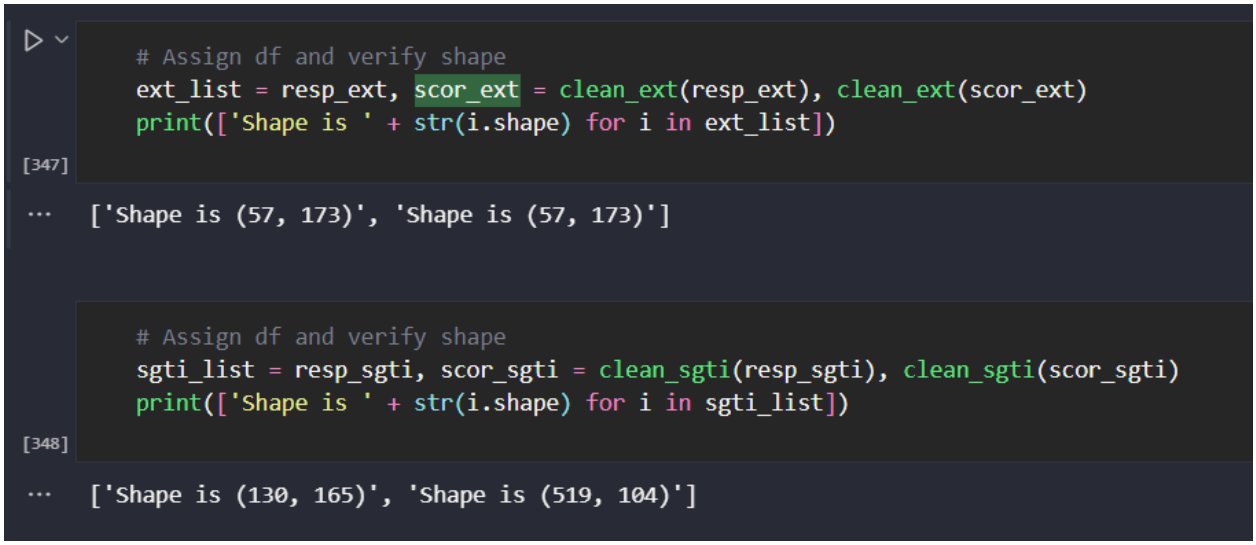

Figure 1: Scoring

To run through the algorithm, the first step involves processing the data from both the Extracted and SGTI sources, and obtaining the shape of the tables (Dataframes). From the figure above, we can see that the shape of the Extracted data is 57 Rows, and 173 Columns, and the line is repeated twice for responses and scores respectively, implying that there are 57 companies obtained using the extraction from the application and a total of 173 questions processed for responses and scoring (The extraction seems to leave the unscored questions as 0, hence why responses and questions are equal). In the following line, we see that for the initial response dataset provided by Aster, we only have sample of 130 companies (it was sufficient enough to cover the 60 companies), and the subsequent listing provided by Zecharias is a complete listing of 519 companies with 104 in scope questions (fewer questions due to scoring).

Thereafter, all four sheets are processed using simple cleaning algorithms and compared using Pandas built-in compare function. While we will not run through the details of the cleaning algorithm in detail, it essentially creates a 1-1 match for questions and companies between the Extracted data and the SGTI data.

```
# Print metrics (Uncomment to see)
print('Number of missing questions: ' + str(len(missing_qns)))
print('Missing questions are: ' + str(missing_qns))
print('Data has ' + str(output.shape[0])+' rows and ' +str(output.shape[1])+' columns')
```

Figure 2: Specific metrics

For specific metrics, the researcher must uncomment the following lines of code which have been commented out for efficiency. These 3 lines of code will highlight missing questions, or questions that are present in the SGTI data but not present in the Extracted data. For missing questions, it will return a list of missing questions. If there are missing companies, there will be a key error with the missing companies as below, this will require manual rectification on the researchers part to edit the excel spreadsheet for naming errors. These specific metrics will be shown in the following section.

3. Issues Detected in Testing

3.1 Company names

```
KeyError: "['BONVESTS HOLDINGS LIMITED', 'EXCELPOINT TECHNOLOGY LTD.', 'HAW PAR CORPORATION LIMITED', 'NET PACIFIC FINANCIAL HOLDINGS LI
MITED'] not in index"
```

Figure 3: Unmatched company names

In testing the 57 companies, the Extracted data contained several companies that were unable to be found in the SGTI data. As the algorithm does a word for word reconciliation, differences in the naming convention will be highlighted. In our case, the following companies were found in the two datasets.

Name in Extracted data	Name in SGTI data
BONVEST HOLDINGS LIMITED	BONVESTS HOLDINGS LTD
EXCELPOINT TECHNOLOGY LTD.	EXCELPOINT TECHNOLOGY LTD
HAW PAR CORPORATION LIMITED	HAW PAR CORP LTD
NET PACIFIC FINANCIAL HOLDINGS LIMITED	NET PACIFIC FIN HLDGS LTD
ALEMBIC 2020	N.A.

Table 1: Unmatched company names

These differences must be rectified manually by editing the spreadsheets as I had done in the attached excel sheet, as it would not be feasible to design an algorithm to detect such sporadic and infrequent instances of unmatched company names.

3.2 Missing questions

```
Number of missing questions: 12
Missing questions are: ['E.2.2.(B)', '31 (a)', '31 (b)', 'A.3.5-A.3.6', 'E.3.14', 'NCG9 (a)', 'NCG9 (b)', 'D.1(b)', 'NCG12', 'NCG13.2',
'Notice', 'AGM']
Data has 57 rows and 153 columns
```

Figure 4: Questions in scope but not picked up by the application

After manually rectifying the issue highlighted in Section 3.1, we run the specific metrics code to find that the questions that are in scope for SGTI based on the provided dataset but are not within the Extracted data are as follows. While it is beyond the scope of testing to check each question on its own, the users of this report and the provided code may use these to highlight issues with the application. In our case, the missing questions are above in Figure 4.

3.3 Matching Errors

After detecting the missing questions, the algorithm does a 1-1 match for companies and questions that are available in both the Extracted data and the SGTI data. While the code can be adjusted to include all companies, it does not present useful information as we will discuss in the limitations section.

```
compare_dfs(resp_ext, resp_sgti, '3CNERGY LIMITED')[0]
```
✓ 0.9s Python

	28	29	32	37	C.1.12	D.6.3	D.8.1	NCG13	67	72	...	76	NCG10	79	RGG02	21	71	B.1.1(B)	C.1.1(B)	RGG03	85
self	3	2	B3	E	N	Y	Y	N	3	0	...	0	3	0.0	0	2	2.0	0	0	0	2.0
other	3	2	Disclosure in Bands	N	0.0	0.0	0.0	Y	Y	N	...	N	Y	N	N	Y	Y	N	N	N	0.0

2 rows × 22 columns

```
compare_dfs(scor_ext, scor_sgti, '3CNERGY LIMITED')[0]
```
✓ 0.4s Python

	3	10	11	12-13	37	A.3.16	A.3.15	B.5.2	D.6.4	NCG13	67	NCG10	21	71
self	3.0	2.0	0.0	0.0	1.0	2.0	2.0	2.0	1.0	0.0	3.0	3.0	2.0	2.0
other	0.0	1.0	10.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0

Figure 5: Case Study on 3CNERGY LIMITED

As such, the design of the algorithm was to accept inputs for the individual company name, in this case "3NERGY LIMITED" will be used as the benchmark. The differences in the responses sheets can be

found in the first line of code, whereas the differences in the scores can be found in the second line of code. If the researcher would like to explore the questions that are unmatched, he may run the code with .columns.tolist() to find the questions that are missing for a particular company, as below.

```
compare_dfs(resp_ext, resp_sgti, '3CNERGY LIMITED')[0].columns.tolist()
✓ 0.9s
['28',
 '29',
 '32',
 '37',
 'C.1.12',
 'D.6.3',
 'D.8.1',
 'NCG13',
 '67',
 '72',
 '73',
 '75',
 '76',
 'NCG10',
 '79',
 'RGG02',
 '21',
 '71',
 'B.1.1(B)',
 'C.1.1(B)',
 'RGG03',
 '85']
```

Figure 6: Demonstrating the use of .columns.tolist()

4. Additional Metrics

In order to gain a sense at a glance of the performance of the application in scoring, and assuming that the manual encoding done by interns is 100% correct, we can rely on “accuracy” of the application. This is obtained by taking the percentage of questions that are an exact match in both Extracted datasets and SGTI datasets, i.e. they are completely correct. This is computed on an individual company basis in the algorithm, shown below in Figure 7.

```python
# Standardising the original df, needs generated ext cols to create a 1-1 match
def compare_dfs(ext_df, sgti_df, company):

    # Obtain comparables using previous function
    previous = parse_dfs(ext_df, sgti_df)
    companies = list(ext_df.index.values)
    final_cols = previous.columns.tolist()
    final_col_len = len(final_cols)

    sgti = sgti_df.loc[companies, final_cols].fillna(0)

    # Comparing the two
    comp = sgti.compare(previous, align_axis = 0)
    comp = comp.loc[company].dropna(how = 'all', axis =1 )
    comp_len = len(comp)

    accuracy = round(100*(1-(comp_len/final_col_len)), 1)

    # print('Accuracy: ' + str(accuracy))
    return comp, accuracy
```

Figure 7: Individual accuracy metrics

For each company, we will obtain the accuracy of the questions and the algorithm will store each accuracy value. As we run our final line of code, we will obtain the average accuracy for all companies in the sample (57 companies).

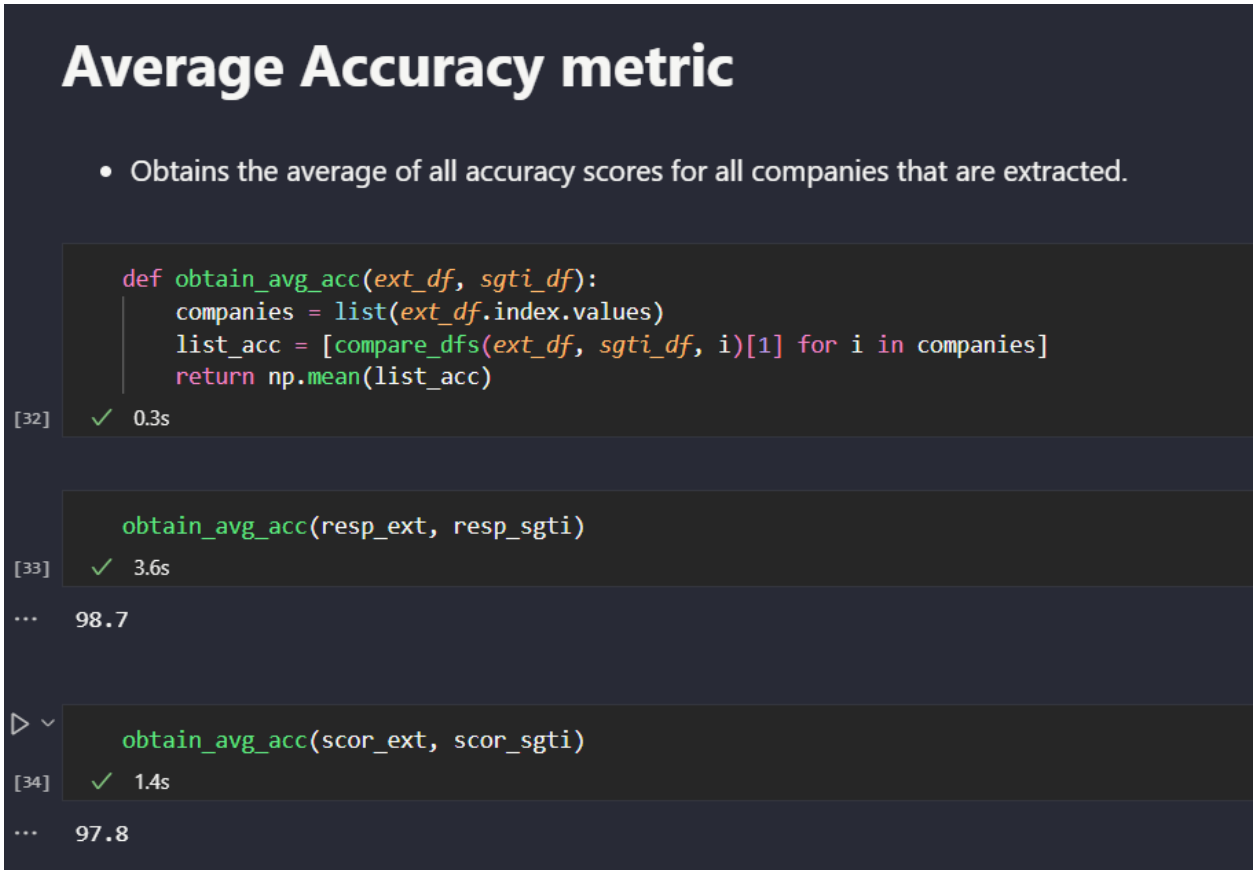

Figure 8: Average accuracy metrics

In our first pass of the accuracy testing, we obtain the results below:

Response or Score	Accuracy
Response	98.7%
Score	97.8%

While I (Tat Sheng) do not have the authority to assess the threshold for accuracy, it is imperative for the main researcher to identify a suitable accuracy threshold for the application to work. If the threshold is insufficient, and the researcher would like to have a 100% accuracy, the researcher can raise this to the TCS to rectify the application based on the findings highlighted in Section 3.

5. Limitations

The largest limitation for this testing process involves the accuracy of the manually encoded data. A fundamental assumption that underpins the accuracy metric and highlights the missing questions, unmatched companies and different datapoints is the fact that the manually encoded data is accurate in the

first place. If the data is 100% accurate, then we may be able to obtain the 100% accuracy score using the algorithm detailed in section 4. However, if the SGTI data is not 100% correct, then the accuracy score above may be restricted to an accuracy that is of a lower level. Thus, based on the researchers intuition of the accuracy of the scoring, it is entirely justifiable for a lower accuracy score to be accepted as correct.

6. Conclusion

To conclude, as the main focus of this report is to detail the testing process on the responses and the scoring of the application, we have run through several metrics we used in the testing process and hope that the findings will be useful, and reusable in the future.

I have had a great time working for the CGS, please feel free to reach out to me for anything.

- Tat Sheng