50.038 Computational Data Science

Project Report

Title: Prediction of Corruption Perception Index of a country based on macro-economic features

Team Name:
Incorruptibles
Clarence Toh - 1003080
Monica Saravana - 1002888
Kang Yun Yi - 1000593

Table of Contents

Abstract and Motivation

Corruption in a country and its government is one of the main evils that hinder its progress. It is common knowledge that most of the developed countries are perceived as corruption free and the developing and under-developed countries have higher rates of corruption. Because of its nature, corruption is a difficult measure to quantify and there a few estimates that manage to do it. One of them is Corruption Perception Index, an estimate that quantifies corruption based on the perceptions of people living in that country by conducting extensive surveys and statistically interpreting the results. In our project, we would like to model these perceptions of people using socio and macro-economic factors to predict the CPI of countries. The results that we obtain would hence, attempt at quantifying and unifying varying perceptions of people from different countries across different times

Goals

We hope to be able to predict future CPI values and attain the specific indicators that induce a poor perception of a country's transparency level.

Dataset

Our dataset consists of socio and macro economic features of about 180 countries. Since were trying to model the perceptions of people about corruption, we included every social and economic variable of a country to see which actually contributed to the perception of corruption, and hence collected over 1080 features across 180 countries present in the CPI. These include financial data of a country such as Net national income, Central government debt, GDP and many others. Some social indicators includes the Birth rate, Ratio of educated females, etc, and even environmental features such as Forest cover depletion and Carbon Dioxide emissions.

The data collected was organised in two ways:

3. All countries dataset - Countries vs Indicators dataset for every year : Dimensions: 180 X 1080 - For 180 countries
4. Time-series dataset - Year vs indicators for a single country: Dimensions: 22 X 1080 - For 22 years

The dimensions vary from one dataset to another because there were some indicators that were not documented for certain countries.

Data Collection

Source

Corruption Perception Index (CPI)
Source: https://www.transparency.org
Format: csv
Description: Yearly (1995 - 2016) scores of each country's CPI values

World Bank World Development Indicators (WWDI)
Source: www.quandl.com
Format: pandas dataframe
Description: Time series of each country's macroeconomic indicator in pandas dataframe eg. a time series of Singapore's GDP from 1995 - 2015

Methodology

As the CPI values were stored in csv files, it was easy to get our hands on the values. Getting WWDI indicators, on the other hand, posed a challenge. We had 180 countries and up to 1082 indicators for each of them that had to be called using Quandl API. If we take the product of 180 and 1082, it would mean that we had to make at most **194,760 calls** to the API server.

Earlier, we aimed to model corruption using macroeconomic indicators only from a single year. We adopted an iterative approach and called these indicators one by one using Google Colab. Turns out it was inefficient and a waste of API calls for we were only storing one row (a particular year) from all the time series that we are calling from Quandl. The entire process took approximately 4 days to complete. This was because for a user's single API key was only allowed to make 50,000 API calls daily. Another reason was that they capped the rate of API calls every 10 minutes. Next, we were unaware of Google Colab disconnecting from inactivity. We ran the program over the weekend and left our computers unattended from time to time thus we had to restart the program from where we left off a few times. Finally, we preprocessed the data ie remove/ filling null values right at the point of calling the API which made the program too bulky and inefficient because we did not back the original time series.

After our first presentation, we decided to improve the existing approach. We wanted to make better use of the data that was available to us and decided to make a CPI prediction for a select few countries. To reduce the bulkiness of the program, it only called and stored the data on Google Drive, all preprocessing will be done later. Lastly, we parallelised the process by opening multiple Google Colabs and created multiple Quandl API keys for each of the Colab we

ran and we managed to shave off a significant amount of run time for the data gathering. The time taken was approximately 16 hours, which is an **83% reduction** from the original run time!

Data Preprocessing

For better understanding our our data we split in into 5 classes based on CPI values. The classes followed as such,

Class 0: CPI score of 0 to 20 : perceived as corrupt
Class 1: CPI score of 21 to 40 : perceived as slightly corrupt
Class 2: CPI score of 41 to 60 : perceived as neutral
Class 3: CPI score of 61 to 80 : perceived as slightly transparent
Class 4: CPI score of 81 to 100 : perceived as transparent

The data collected had a lot of missing and null values which had to be addressed. Replacing null values by 0 or mean/median of the column would make a significant statistical impact because we have countries from different levels of corruption in the dataset and this might skew the results we get. Therefore, we decided to split the data into its respective classes, and drop columns with more than 30% missing values. By doing this, we reduce the number of features that we have and ensure that whatever value we use to choose to replace the remaining 29% missing values with (median in our case), would not have a significant statistical impact to the results.

This stage of preprocessing uncovered some interesting observations. Since we ran this for separate classes, we observed that the columns with over 30% missing values in each bin never overlapped. This means that columns with missing values in one class, always had values in the other classes. Since data transparency is one of the major indicators of corruption, it was interesting to observe that missing values of a particular class of corruption did not overlap with other classes.

The preprocessing on the Time Series dataset, also followed a similar approach of drooping columns with over 30% missing values and replacing the remaining nulls by column median.

Data Exploration

After dropping columns with missing values, we deduced that there would be a high correlation between columns as the data had multiple columns with values in similar units but under different scenarios. For example, one indicator would be Net Bilateral Aid Flow (total), which is the amount of Aid received, that would be positively correlated with Net Bilateral Aid Flow (Japan) and Net Bilateral Aid Flow (US) and the other countries contributing to it.

Hence the correlation matrix for 776 features looked like Fig 1. With several blue and red spots indicating high positive and negative correlation.

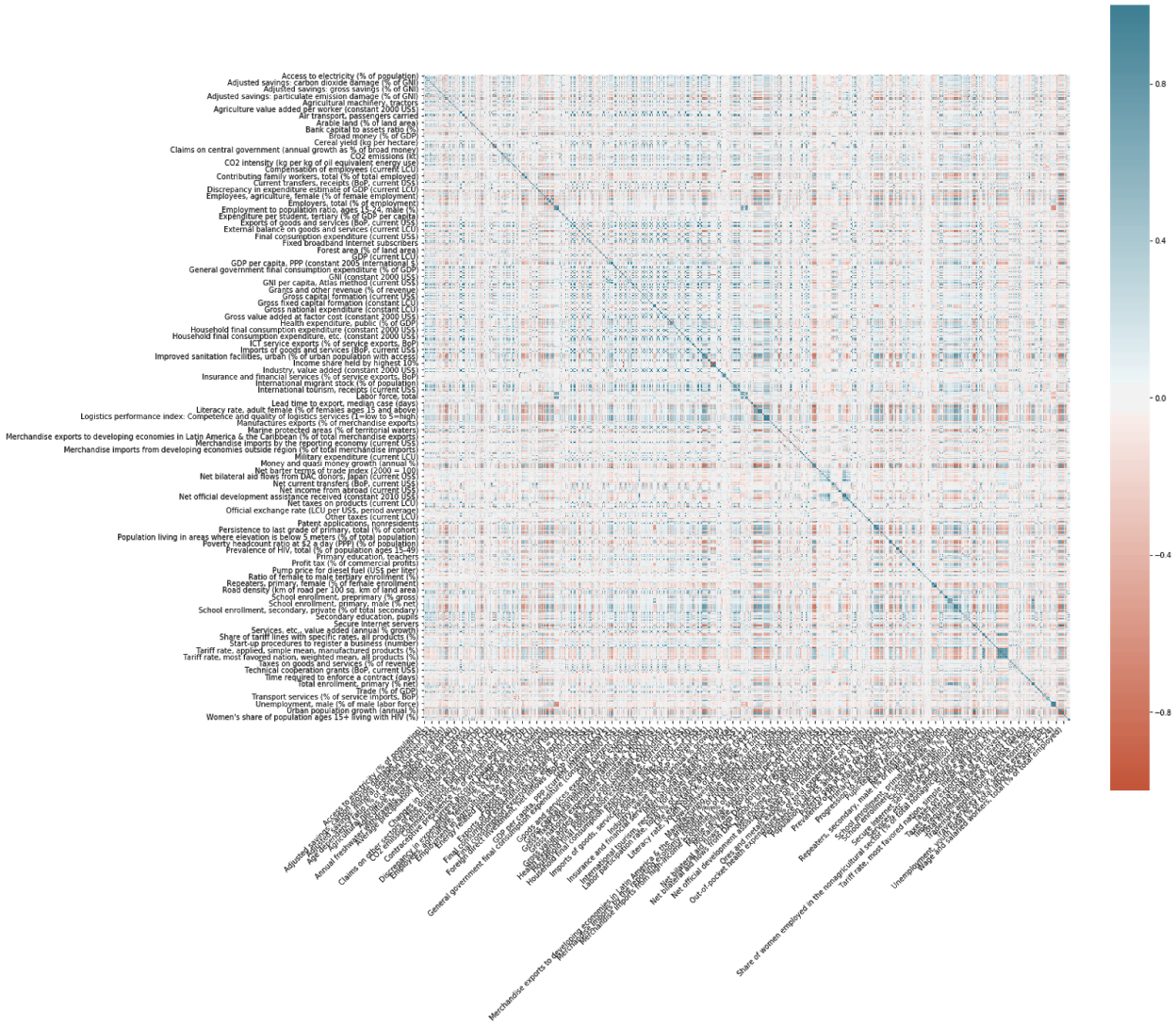

Fig 1: Correlation Matrix

In the All Countries dataset we also observed that we have more data corresponding to CPI values in the range 21 to 40 than the rest of the data. When we separate them into bins and try to solve a classification problem, this imbalance in data might cause skewness in the results. In general, we had more data for countries that fall under class 1 than the others.

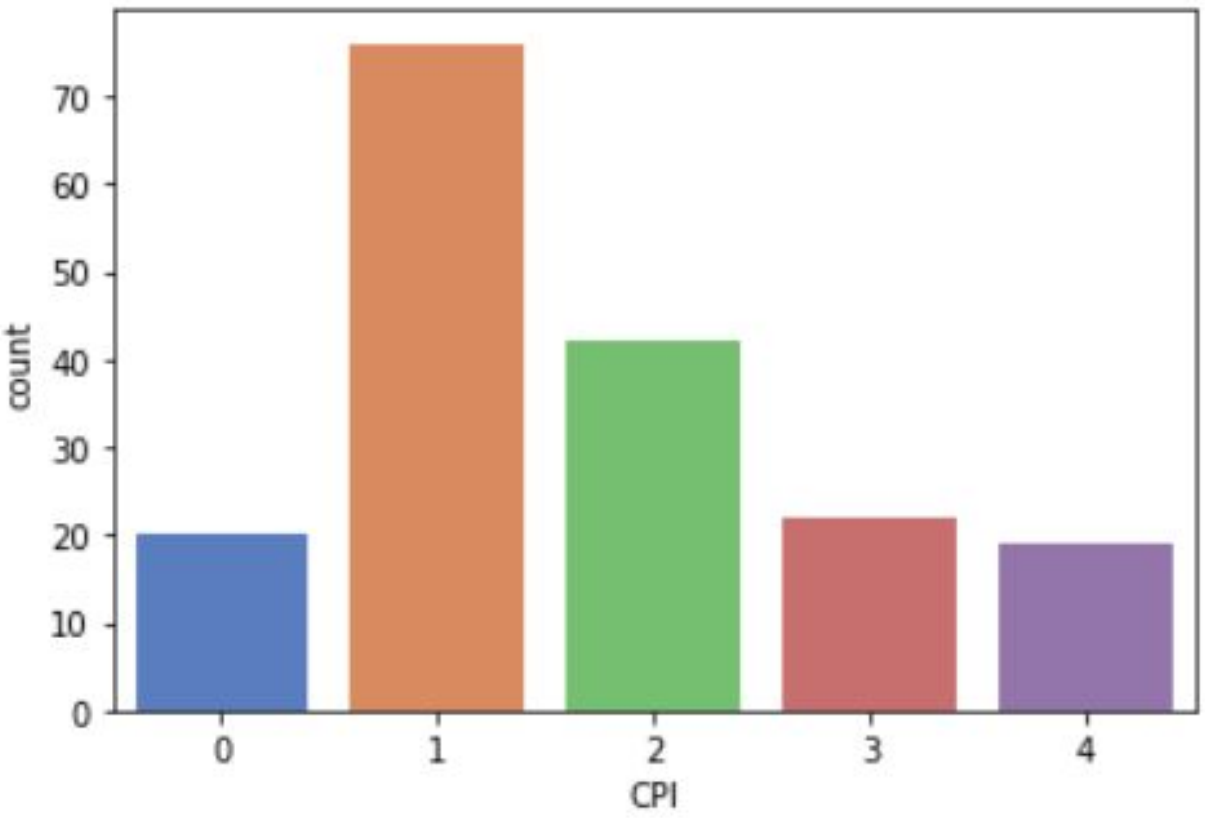

Fig 2: Graph indicating the imbalance in data

Initial Approach: Modelling the All-Countries Dataset

In order to solve the problem at hand, the first approach was to tackle it as a multi-class classification problem. The goal would be to use the All Countries dataset for one year and use classification algorithms to categorise a new country with a given number of features into one of the 5 classes described.

Dimensionality Reduction

After the initial pre processing, the dimension of the dataset was 180 countries X 776 Indicators.

In order to improve the shape of this dataset we explored the correlation of variables further. The data showed that there were 309 column pairs among 776 columns that had correlation greater than 0.8. This would be a problem while performing analysis later as it induces multicollinearity into the dataset. Hence, we dropped one of the columns of the 309 pairs while retaining the other in our dataset. The resultant correlation matrix looked like Fig.1, with

columns that still show perceivable levels of positive correlation (indicated in blue) and negative correlation (indicated in red), but whose values <= 0.7

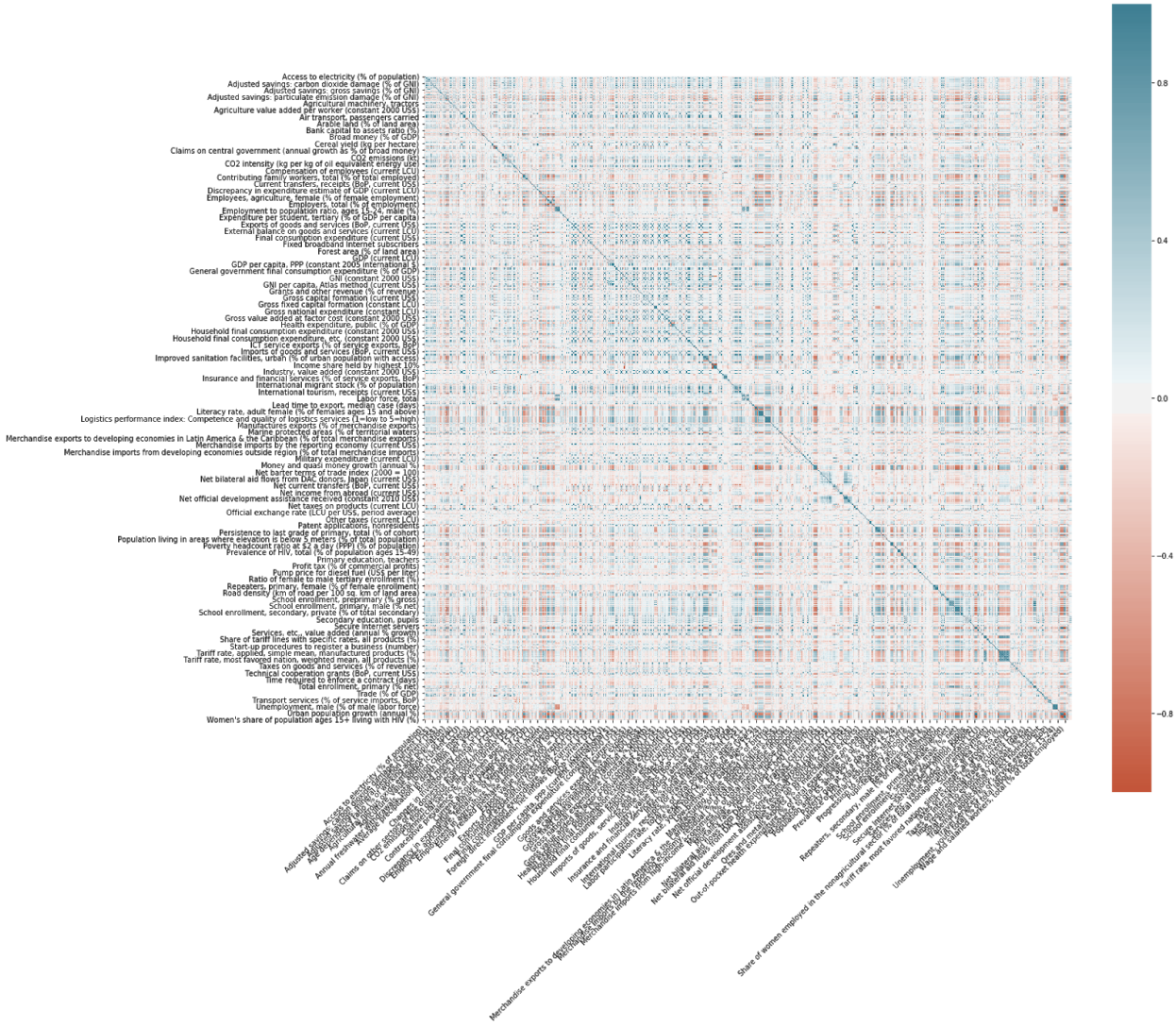

Fig 3: Correlation matrix after dropping highly correlated variables

Even after dropping columns with high percentage of missing columns and correlation, there were 467 indicator columns which we would use to model CPI. We tried to perform some means of dimensionality reduction to improve the shape of our dataset. Due to the nature of our problem statement and the goals we set for ourselves, we wanted did not choose to do PCA or t-SNE as those methods combine multiple columns to one, making the output less meaningful than the original data.

Hence, a couple of feature selection models were applied to see what our top features were and if we could eliminate a few of our variables. Since we were tackling a classification problem we chose models that best suited it,

1. Extra trees classification:

```
extra_tree_forest = ExtraTreesClassifier(n_estimators = 10, criterion
='entropy', max_features = 2)
```

This method makes the use of splitting the data at different nodes with randomly selected features to compute feature importances. The feature importances were normalized between 0 and 1 and from Fig 3, we observe that even the most important feature did not hold a very significant weight.

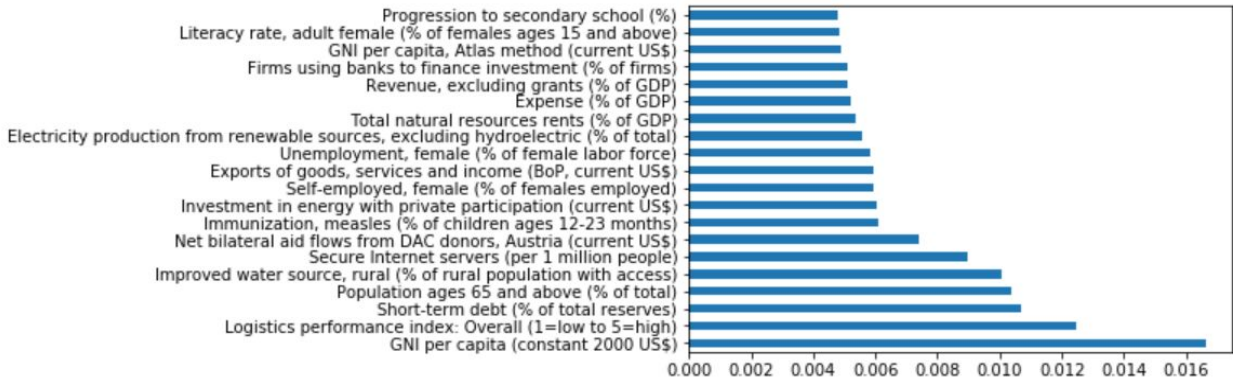

Fig 3: Top 20 features according to Extra Trees Classification

2. Chi-squared:

```
best_features = SelectKBest(score_func=chi2, k=200)
```

This method calculates the dependency of one feature on the response variable by making use of the difference between the observed and expected value of the features when distributed according to Chi-squared distribution. The feature importances showed were more significant compared to Extra trees classifier and the top 20 features according to this model are displayed in Fig: 4

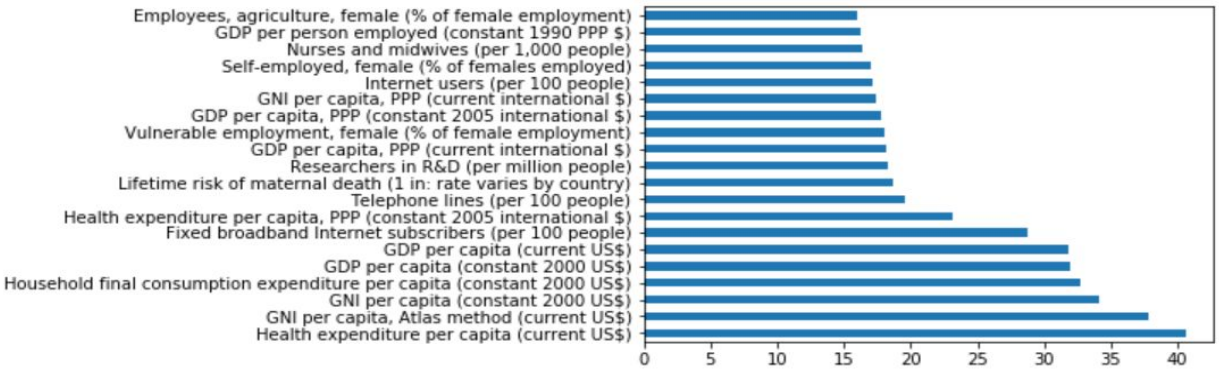

Fig 4: Top 20 features according to Chi-squared Classification

The main challenge with these methods is that the output is in terms of rankings of features and a list of their feature importances. There is no clear way of determining what the optimal number of features are and that would have to be computed based on trial and error.

3. Recursive Feature Elimination:

```
model = LogisticRegression(solver = 'lbfgs', multi_class='multinomial',
max_iter = 1000)
```

We ultimate settled on using Recursive Feature Elimination using logistic regression to give us the optimal number of features. The results were as such:

```
Optimum number of features: 53
Score with 53 features: 0.490566
```

While the model did well in eliminating a lot of features, the accuracy of logistic regression was something that needed to be improved on. Since we had 5 classes, even a model that pick a class at random would have 0.2 expected accuracy. Our model does perform slightly better with an accuracy of 0.49 and this could be considered our baseline model while trying other models to better this accuracy.

The top 53 features selected by this method are:

Financial	Social	Environmental
'Net official flows from UN agencies, WFP (current US$)',	'Access to electricity (% of population)'	'Plant species (higher), threatened', 'Physicians (per 1,000 people)',
'Net official flows from UN agencies, UNTA (current US$)',	'Net intake rate in grade 1 (% of official school-age population)',	'Other greenhouse gas emissions, HFC, PFC and SF6 (thousand metric tons of CO2 equivalent)',
'Net official flows from UN agencies, UNFPA (current US$)',	'Population, female (% of total)',	'Ores and metals imports (% of merchandise imports)',
'Net official flows from UN agencies, UNDP (current US$)',	'Population living in areas where elevation is below 5 meters (% of total population)',	'Ores and metals exports (% of merchandise exports)',
'Net ODA received per capita (current US$)', 'Net migration',	'Population growth (annual %)',	'Marine protected areas (% of territorial waters)'
'Net income (BoP, current US$)',	'Population density (people per sq. km of land area)',	
'Net current transfers from abroad (current US$)',	'Persistence to last grade of primary, female (% of cohort)',	
'Net current transfers (BoP, current US$)',	'Persistence to grade 5, female (% of cohort)',	
'Net capital account (BoP, current US$)',	'Permanent cropland (% of land area)',	
'Net bilateral aid flows from DAC donors, United States (current US$)',	'Part time employment, female (% of total female employment)',	
'Net bilateral aid flows from DAC donors, United Kingdom (current US$)',	'Out-of-pocket health expenditure (% of private expenditure on health)',	
'Net bilateral aid flows from DAC donors, Total (current US$)',	'Other taxes payable by businesses (% of commercial profits)',	
'Net bilateral aid flows from DAC donors, Spain (current US$)',	'Other taxes (% of revenue)',	
'Net bilateral aid flows from DAC donors, Norway (current US$)',		
'Net bilateral aid flows from DAC donors, Netherlands (current US$)',		
'Official exchange rate (LCU per US$, period average)',		
'Net bilateral aid flows from DAC donors, Japan (current US$)',		
'Oil rents (% of GDP)',		
'Portfolio investment, excluding LCFAR (BoP, current US$)',		
'Portfolio equity, net inflows (BoP, current US$)',		
'Other expense (% of expense)',		
'Net bilateral aid flows from DAC donors, Italy (current US$)',		
'Net bilateral aid flows from DAC donors, Germany (current US$)',		
'Net bilateral aid flows from DAC donors, France (current US$)',		
'Merchandise exports to economies in the Arab World (% of total merchandise exports)',		
'Merchandise exports to developing economies outside region (% of total merchandise exports)',		
'Merchandise exports to developing economies in Sub-Saharan Africa (% of total merchandise exports)',		
'Merchandise exports to developing economies in South Asia (% of total merchandise exports)',		
'Merchandise exports to developing economies in Middle East & North Africa (% of total merchandise exports)',		
'Merchandise exports to developing economies in Latin America & the Caribbean (% of total merchandise exports)',		
'Merchandise exports to developing economies in Europe & Central Asia (% of total merchandise exports)',		
'Merchandise exports to developing economies in East Asia & Pacific (% of total merchandise exports)',		
'Merchandise exports by the reporting economy, residual (% of total merchandise exports)',		

CPI is a value calculated based on surveying experts in various fields and commoners alike. It is interesting to observe that the top features selected by our RFE model also reflect this as there is a fair distribution of features in the financial, social as well as environmental sectors.

Modelling

The All countries dataset was now of shape 180 X 54, with 180 countries' data containing 53 most important features and respective CPI classes. We ran a couple of classification models to check their performance.

The imbalanced dataset gave us mediocre results, and we observed that the results were skewed to class 1 as there were more samples from there.

```
[[ 0  4  0  0  0]
 [ 0 13  5  0  0]
 [ 0  3  4  0  0]
 [ 0  0  4  0  0]
 [ 0  2  1  0  0]]
Accuracy: 0.4722222222222222
```

Fig 4: Confusion matrix for Random Forest Classifier

```
[[ 0  4  0  0  0]
 [ 0 14  4  0  0]
 [ 0  3  4  0  0]
 [ 0  3  1  0  0]
 [ 0  3  0  0  0]]
Accuracy: 0.5
```

Fig 5: Confusion matrix for K nearest neighbor classifier

Most of the other classifiers we learnt such as SVM, Naive Bayes classifiers and simple Decision trees also faced a similar issue.

In order to overcome the imbalance, we considered adopting some upsampling techniques such as Bootstrapping and SMOTE. However, we decided against using them because replicating samples or resampling by knn means by each class for 53 features causes redundancy of data. This is problematic as it becomes easy for the model to pick up trends because of the replicated data and gives unnaturally high scores for accuracy. This may also lead to overfitting of the data.

In order to overcome the imbalance without having to resort to upsampling, boosting based methods was applied to our data. These methods deal with class imbalance by constructing successive training sets based on incorrectly classified examples.

One such method was Adaptive Boost Classifier which gave an accuracy score of 0.4775283188792453.

```
abc = AdaBoostClassifier(n_estimators=1500,learning_rate=1.3)
```

We also tried Adaptive Boosting with an SVC as base estimator which gave an accuracy of 0.4528301886792453.

```
svc=SVC(probability=True, kernel='linear')
abc = AdaBoostClassifier(n_estimators=100, base_estimator=svc,learning_rate=1)
```

Since none of our models performed better than baseline logistic regression model, we decided to move to a time series approach to check if it can give us better results.

Second approach: Modelling the Time-Series dataset per country

Time-Series dataset has time series values of the indicator variables. Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) in this case, address the issue of sequential data. Long-Term Dependencies are ignored because the statistics 20 years ago do not give us any insight about what's going to happen in the next year. However, the last two to three years may give us some insights, if nothing drastic happens. In such cases, where the gap between the relevant information and the place where it is needed is small, RNNs can learn to use the past information.

The data frame of a country looks like:
In this case the country for experimentation is Australia. The dataframe shows the top 5 rows, however the entire dataframe has 22 rows which means that there are 22 years of data. In contrast, for countries such as China, there is data for 14 years only.

	Date	AUS_TM_TAX_MANF_BR_ZS	AUS_TM_TAX_TCOM_BR_ZS	AUS_FM_LBL_BMNY_GD_ZS	AUS_FM_LBL_BMNY_CN	AUS_FM_LBL_BMNY_ZG
0	1996-12-31	11.83	2.5	62.896067	3.318390e+11	10.648952
1	1997-12-31	11.83	2.5	64.128246	3.560984e+11	7.310580
2	1998-12-31	11.83	2.5	65.655622	3.861187e+11	8.430358
3	1999-12-31	11.83	2.5	69.588639	4.313111e+11	11.704258
4	2000-12-31	11.83	2.5	67.706080	4.474417e+11	3.739910

5 rows × 850 columns

Fig 6: First 5 rows and first 5 columns of the dataset for Australia

AUS_SH_STA_BRTC_ZS	AUS_SP_DYN_CBRT_IN	AUS_EN_BIR_THRD_NO	AUS_TM_TAX_MRCH_BR_ZS	CPI
99.3	13.9	NaN	9.69	86.0
99.3	13.6	NaN	9.69	88.6
99.3	13.3	NaN	9.69	87.0
99.3	13.1	NaN	9.69	87.0
99.3	13.0	NaN	9.69	83.0

Fig 7: Last 5 rows and first 5 columns of the dataset for Australia

The dimensions are [number of years x number of columns].

The preprocessing was done as described earlier, by removing columns which has more than 30% null values, and replaced the remaining cells with the median of the column.

Dropping columns based on high collinearity seemed inappropriate for Time Series modelling as a good model would identify the correlation and analyse trends based off it and hence we would lose data by the removal of collinear columns.

Thus, our approach for applying a Time Series based prediction model, was to pick the Time Series dataset for one country in every class of corruption, and to build a specific model to analyse the outcome and results.

Our naive LSTM model took 50 units, with relu as the activation function, a single dense layer of 1 and 5 as the number of steps to take. For this annual time series, it yielded a 20.44 percentage error. Looking at this, we felt that the result is too far off what we wanted. Our next approach was to interpolate the data as we suspected that the data set was too small. Hence we interpolated in a linear fashion to create monthly and daily data. The data is normalized using the minmax scaler from 0 to 1. The time step used was 5.

Strategy (only normalizing)	**Percentage error**
Interpolate by year, 22 data points	20.441774368286133
Interpolate by month, 262 data points	1.0591048002243042
Interpolate by day, 7920 data points	0.015229816548526287

Fig 8: Percentage errors after normalizing and interpolating data

The percentage error was greatly reduced by a great distance. However, we believed it was because the data was inflated and very obvious trends were artificially created through interpolation. This led us to our next approach, which was to create a dataframe of percentage changes in relation to the yesteryear.

Strategy (percentage change w/o normalizing)	**Percentage error**
Interpolate by year; number of steps = 7	351430.6875
Interpolate by year; number of steps = 5	320606.21875

Fig 9: Percentage errors after interpolating data of percentage changes without normalization

The results were horrendous. After witnessing shocking results, the dataframe gave off some hints where large percentage changes within years may have influenced the RNN's ability to learn of any trends. In order to eradicate such behaviour, we normalize the data frame excluding the dependent variable. The data is normalized using the minmax scaler from 0 to 1. The method ignores the interpolation to create daily data points as it would improve the percentage error but not the accuracy of the prediction.

Strategy (percentage change w/ normalization)	**Percentage error**
Interpolate by year; number of steps = 7	0.6501209735870361
Interpolate by year; number of steps = 5	0.5464288592338562
Interpolate by year; number of steps = 3	1.6997615098953247
Interpolate by month; number of steps = 7	1.4450266361236572
Interpolate by month; number of steps = 5	0.217756450176239
Interpolate by month; number of steps = 3	0.15020249783992767

Fig 10: Percentage errors after interpolating data of percentage changes with normalization

The percentage errors above produced very good results. Perhaps the hypothesis that sporadic large and small values across the data frame could have posed as a negative influence to the RNN's ability to learn of major trends is true.

Next, in an attempt to bluntly achieve a better model, we decided to classify the percentage changes in the dataframe.

The classification is done in three ways listed below:

1) Negative percentage change will be classified as -1;
   Percentage change of zero will be classified as 0;
   Positive percentage change will be classified as 1
2) Percentage change of zero and negative values will be classified as 0;
   Positive percentage change will be classified as 1
3) Negative percentage change will be classified as 1;
   Percentage change of zero will be classified as 2;
   Positive percentage change will be classified as 3

The method does not involve normalization.

Strategy (classifying the percentage changes as -1,0,1 w/o normalization)	**Percentage error**
interpolate by year; number of steps = 7	63.047935485839844
interpolate by year; number of steps = 5	59.16572189331055
interpolate by year; number of steps = 3	82.13848114013672
interpolate by month; number of steps = 7	11.621817588806152
interpolate by month; number of steps = 5	11.747268676757812
interpolate by month; number of steps = 3	15.268765449523926

Fig 11: Percentage errors after interpolating data of classes (-1, 0, 1) without normalization

Strategy (classifying the percentage changes as 0,1 w/o normalization)	**Percentage error**
interpolate by year; number of steps = 7	1.3483071327209473
interpolate by year; number of steps = 5	2.037656784057617
interpolate by year; number of steps = 3	4.179565906524658
interpolate by month; number of steps = 7	0.44504478573799133
interpolate by month; number of steps = 5	4.9152421951293945
interpolate by month; number of steps = 3	3.9173173904418945

Fig 12: Percentage errors after interpolating data of classes (0, 1) without normalization

Strategy (classifying the percentage changes as 1,2,3 w/o normalization)	**Percentage error**
interpolate by year; number of steps = 7	5.180117607116699
interpolate by year; number of steps = 5	4.566781520843506
interpolate by year; number of steps = 3	1.8392828702926636
interpolate by month; number of steps = 7	3.652085065841675

interpolate by month; number of steps = 5	4.432069778442383
interpolate by month; number of steps = 3	0.7650640606880188

Fig 13: Percentage errors after interpolating data of classes (1, 2, 3) without normalization

The results above were mixed. In case 1, the classification involved -1 , 0 and 1 as classes. The results were not ideal. Applying the classification on the data frame of annual percentage changes achieved a percentage error of 59% at best. This number reduced when applied on the interpolated monthly data, achieving a minimum percentage error of 11.62%.
However, when classification was applied in a binary fashion in case 2, the model seemed to be better at predicting, achieving a minimum percentage error of 0.44%.
Nevertheless, when case 3 was conducted, the model performed in a similar fashion as in case 2, achieving a percentage error of 0.76% at best.

Looking through the past iterations of the LSTM model, we identified that converting the monthly interpolated data to one of percentage changes with a timestep of 3 and the application of a minmax normalization yielded the best results. Taking this result forward, we tuned with several hyperparameters to improve the model, namely the hidden units and adding more layers of the dense class.

Strategy (toggling with the units, and adding more layers)	**Percentage error**
25 hidden units	0.4332916736602783
50 hidden units	0.15020249783992767
75 hidden units	0.08212704956531525
Adding an additional layer with the dense class, 35 units	0.025785423815250397
Adding an additional layer with the dense class, 25 units	0.09989678114652634
Adding an additional layer with the dense class, 15 units	0.016958503052592278

```
model = Sequential()
model.add(LSTM(75, activation='relu', input_shape=(n_steps, X.shape[2])))
model.add(Dense(15))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

The final model looks like this. The percentage error achieved by this model is 0.017% which is the lowest among all other models.

Weights for each layer (Australia as an example):

```
print(warr.shape) # warr is a numpy array of weights for inputs
print(uarr.shape) # uarr is a numpy array of weights for hidden units
print(barr.shape) # barr is a numpy array of bias
(674, 300)
(75, 300)
(300,)
```

RNNs has the ability to model short term dependencies. This is due to the hidden state in the RNN. It retains information from one time step to another flowing through the unrolled RNN units. In this case, we are able to produce the shapes of arrays of weights for inputs hidden units which we declare in our model as well as the biases.

Application of LSTM on countries of other CPI classes:

Country / CPI class	Percentage error
Nigeria (perceived as highly corrupt)	0.019284656271338463
China (perceived as corrupt)	0.18492698669433594
Israel (perceived as neutral)	0.1837015151977539
Australia (perceived as transparent)	0.016958503052592278
Singapore (perceived as highly transparent)	0.07403237372636795

```
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)
```

A seed was set to get reproducible results with Keras.
When performing LSTM on other countries, the predictions was extremely close to the actual values.

Future Improvements

- Corruption remains a widespread global concern, an issue in the research lies in the limitation of data. The data that we chose is a public dataset on socio and macroeconomic indicators released annually. There are many other public datasets that could have been added. The private datasets would have many sensitive but important features that would contribute to our analysis. However gaining access to such information is difficult. Hence, the model and the resulting proposed actions may have been generalized.
- The models rely heavily on the data provided. Data of different countries are different. Some may have more indicators present, some may have more years on record. The culling does remove variables where not enough data is found, but this may also indicate that there is possibly corrupt practices of avoiding the release of data to the public by the governments themselves. Also, such data may be difficult to collect or have been overlooked by the data collection team at the Organisation for Economic Co-operation and Development. As such, LSTM models are different for each country due to the differences in datasets.
- The models we created are good at predicting the results close to the actual value. However, if a dataset of a longer time series could be obtained, it would be interesting to compare if the long term dependencies will be ignored by the model. The data would also not have any interpolation involved to keep the data in its original form.
- As we were satisfied with the results we attained, the tuning of the hyperparameters only considered a few changes. Perhaps, another way to attain the best model would be to optimize this process, by taking the model that produces results with the smallest percentage error by varying the number of layers, hidden units, loss functions and other hyperparameters.
- Instead of predicting the next time step, we could train our model to predict multiple time steps

Final Thoughts

We believe that corruption must be detected through prediction as soon as possible in order for any government to take corrective and preventive measures. Due to the limited resources many countries allocate for combating corruption, efforts should focus on areas most likely to be involved in corruption cases. CPI values in general might not be the best measure for Corruption because they are based on the perception of people and therefore, there is already a certain bias in the data as people of a country get access to the information the government

releases to them. Hence our research has provided us with a reasonable conclusion that LSTM models are able to predict CPI values given major macro economic, social and environmental factors and our feature selection models may allow us to round down to the most important areas where the governments should look into. This can be considered as an early warning system for corruption and can help narrow the governments' focus and better implement preventive and corrective policies.

The methods in this research have often been used to predict financial stock prices or to forecast the weather, but there are not many studies that uses neural networks to predict public corruption. Perhaps, the research in this paper can provide a naive model that could serve as a proof of concept for other research aimed at fighting corruption with machine learning.