EMATM0065 Introduction to AI and Data Analytics Coursework

Student Number: 2399659

2024/05/28

1 Taiwanese Bankruptcy Prediction (max 15%)

1.1

Our data is taken from the Taiwan Economic Journal. The first attribute of this dataset is the class label. Label 1 indicates a company is bankrupt and label 0 indicates the company is not bankrupt. The other features include a variety of financial metrics There are 96 columns and 6819 rows in total. In this study, we use this dataset to thoroughly investigate our algorithms.

1.2

We make the comparison between a deep-learning method and a non-deep learning methods in addressing this binary classification problem. We chose K-Nearest Neighbors(KNN) classifier as our non-deep learning algorithm and simple feedforward neural network (multi-layer perceptron) with an input layer, one hidden layer, and an output layer. The KNN classifier is a simple, instance-based learning algorithm that classifies a data point based on the majority class among its k-nearest neighbors.Its hyperparameters are Number of neighbors (k) and the distance metric. Accuracy score is a suitable evaluation metric for KNN classifier. In the deep-learning model, there are three layers. The Input Layer takes the financial metrics as input features. The Hidden Layer is further divided into two sub-layers first Layer, which is a dense layer with 128 neurons and ReLU activation function to introduce non-linearity. A dropout layer with a rate of 0.2 to prevent overfitting by randomly dropping 20% of the neurons during training. Then after the hidden layer comes the output layer which is a dense layer with a single neuron and sigmoid activation function for binary classification.

1.3

We thoroughly test and train the two models and compare their performance. After building and compiling both models and making sure they are working properly on our dataset, we perform crossvalidation procedure to evaluate the performance of two different models. In this procedure, we mainly split the dataset into 5 folds and for each fold we train and test our model, and compute the accuracy scores. For each model, we store the accuracy scores in a list. Then we compute the mean and standard deviation of accuracy scores for each model and then compare them. Here is a screenshot of my mean accuracy score and standard deviation: From the screenshot above, we can say the scores are very

```
knn: mean=0.967003418799741, sd=0.004073607165024685
nb: mean=0.9665629982948303,sd=0.004471069577435156
```

Figure 1: Screenshot of the results.

similar and there is not much difference between the accuracy score and standard deviation. We plot a box plot to better visualise our accuracy scores for both models: Then to further confirm the fact about the performance of the model we carry out hypouthesis testing using paired t-test and the results are shown below: p-value is greater than 0.05, so we fail to reject the null hypothesis. Therefore, we conclude that there is insufficient evidence to suggest that there is a statistically significant difference in the performance of the models. This is in line with the scores and the box plot. But one could expect

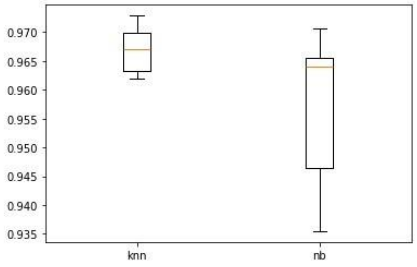

Figure 2: Screenshot of the results.

```
TtestResult(statistic=0.4660360091659996, pvalue=0.6654361834110633, df=4)
```

Figure 3: Screenshot of the results.

the deep learning method to perform better than the KNN, and indeed if you notice in the diagram, the median of nb is quiet closer to the median line representing the 50th percentile is higher in nb than knn. So, I decided to increase the layer of neurons and perform another test. The number of neurons are kept same at 128 but each time another layer of layer of neurons are added. At the time, 2 layers of neurons are added, we noticed that the value of deep neural network rose in the box plots and the median was rising and getting closer to the deep neural KNN's median. At the same time, the p-value decreased. At the third layer of neurons,the deep-nueral network values were greater than knn, This is demonstrated in the below diagram: p-value is less than 0.05, so we reject the null hypothesis. p-value is less than 0.05,

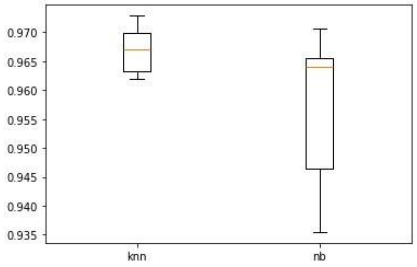

Figure 4: Screenshot of the results.

so we reject the null hypothesis. Details of this test are given in the lab codes as we have a limitation on this. Therefore, we conclude that there is evidence to suggest that there is a statistically significant difference in the performance of the models. Therefore, they are different and the deep neural network with three layers of 128 neurons performs better than the knn model. This also indicates that as we increase the layers of neuroon in a neural network, the performance increases. This aligns perfectly with expectations.

2 Twitter Financial News Sentiment Classification (max 30%)

2.1

Sentiment analysis is about extractings sentiments in text data and identifying it as positive, negative or neutral. It is basically useful for analyzing the opinions of people on various topics and predicting whether these opinions are bearish, bullish or neutral. We chose logistic regression as our model for this sentiment analysis. Logistic regression is a statistical method used for binary classification that can be extended to multiclass classification. It models the probability that a given input belongs to a particular class using the logistic function. In the context of sentiment analysis, logistic regression is used to predict whether the sentiment of a tweet is Bearish (0), Bullish (1), or Neutral (2). LR is estimating the parameters or the

coefficient in linear combination. There are two variables one independent variable and the other dependent variable. Like every other model, logistic regression has its own strength and limitations. First of all, Logistic regression is easier to implement, interpret and very efficient to train. It makes no assumptions about distributions of classes in feature space. It provides a good baseline method to be compared against other models possibly more advanced or complicated models. The limitations of logistics regression is that it constructs linear boundaries and so it assumes a linear relationship between the features and the log odds of the classes. This might not capture complex patterns in the data. It might not be good in tackling linguistic complexities like sarcasm, iron and nuanced expressions in text. It has challenges in keeping pace with lingustic evolution and confronting noisy data. We load both our training and test datasets. The first step is to do good data pre-processing. We define a custom tokenizer that uses NLTK's WordNetLemmatizer to lemmatize tokens. Lemmatization reduces words to their base or root form. It does this lemmatizing each token three times, first as a noun, then as a verb and finally as an adjective. The CountVectorizer is initialized with the custom LemmaTokenizer. This will tokenize the text, lemmatize each token, and then vectorize the text into a bag-of-words representation. This means each sentence is now a list of words it contains without the ordering. Learns the vocabulary from the training data.CountVectorizer with a custom lemmatizing tokenizer to process your text data. This setup should effectively handle basic text preprocessing and feature extraction. So tokenization, lemmatization and Vectorization are our pre-processing steps. Tokenization involves his involves splitting the text into individual words or tokens. Lemmatization involves Reducing words to their base or root form. Vectorization involves converting text into numerical features. Then we test and train our model and print the classification report to see the scores. Then we carry out further studies to determine if the addition of lexicon features improves performance and also if using bigrams in our does lead to an improvement in performance. These are our ways to see if we can further improve our model. The features I used are Unigrams (Single Words), Bigrams (Pairs of Words) and Lexicon-Based Features like VADER. Unigrams capture the occurrence of individual words in the text, which can be highly informative for sentiment analysis. Bigrams capture some context by considering word pairs, which can provide additional information beyond unigrams. Lexicon-based features provide sentiment-specific insights that are not captured by simple word frequency. They help in understanding the sentiment conveyed by the text. Hpotheses are that Proper tokenization and lemmatization will improve model performance by reducing noise and dimensionality, leading to more consistent and meaningful features. Unigrams will provide a solid baseline for text representation, capturing essential information about the presence of important words. However, they might miss context that could be captured. including bigrams can enhance model performance by capturing some context and relationships between consecutive words that unigrams miss. This can be particularly useful for understanding phrases and idioms. Adding sentiment scores from a lexicon like VADER will improve performance by providing additional sentiment-specific information. These features can help the model understand the overall sentiment of the text, especially if the dataset contains a lot of sentiment-related information.

2.2

We used software libraries. These are NLTK, Scikit-learn, *load$_d$ataset*, CountVectorizer, *word$_t$okenize*, WordNetLemmatizer, LogisticRegression, sklearn.metrics, matplotlib, seaborn, RandomForestClassifier, SVC, hstack. The dataset is already divided int training and testing sets. O loaded the training and test data using the *load$_d$ataset* library. We transform our train and test data into vectors using the LemmaTokenizer followed by CountVectorizer method. Then we fit the model using the training data and training lables, and then after fitting we predict the data on testing data to get the predicted labels. The accuracy score is achieved by comparing the original test labels with the predicted labels, and measuring the accuracy score. We present our results using confusion matrices, showcasing the model's performance in classifying each sentiment category. Additionally, we provide bar plots illustrating the distribution of predicted sentiment labels.The confusion matrices reveal that the model performs well in predicting positive sentiment but struggles with neutral and negative sentiments. This suggests a potential imbalance in the dataset or challenges in distinguishing between neutral and negative expressions. To test our models we also trained and tested other models and compared their accuracy scores. We compared the accuracy scores of Logistic

regression with Random Forest classifier and Support Vector classifier, both are very well known models. This gives us indication how our model performs compared to other models. We also tried to implement some hyperparameter tuning code. In our lab file, we have discussed each every step with texts and comments.

2.3

We selected accuracy, precision, recall, and F1 score as they provide a comprehensive evaluation of our sentiment analysis model's performance across multiple dimensions. However, it's essential to note that accuracy may be misleading in the presence of class imbalance. We present our results using confusion matrices, showcasing the model's performance in classifying each sentiment category. This is shown below: Additionally, we provide bar plots illustrating the distribution of predicted sentiment labels. The figure below shows our coonfusion metric: True Negatives are 1840 instances, predicted positive

```
[[1840 1780  352]
 [ 824 4251  862]
 [ 136  940 1299]]
```

Figure 5: Screenshot of the results.

are 1780 instances, Predicted neutral are 352 instances. The model correctly identifies 1840 instances of negative sentiment. However, it wrongly classifies 1780 instances of negative sentiment as neutral and 352 instances as positive. The high number of false positives indicates a tendency to misclassify negative sentiment as either neutral or positive. The model correctly identifies 4251 instances of neutral sentiment. However, it fails to capture 824 instances of neutral sentiment, misclassifying them as negative. Additionally, it incorrectly classifies 862 instances of neutral sentiment as positive. The model correctly identifies 1299 instances of positive sentiment. However, it fails to capture 136 instances of positive sentiment, misclassifying them as negative, and 940 instances as neutral. The classification report is shown below:

```
Accuracy = 0.6015955714750896
Precision (macro average) = 0.5946223310187588
Recall (macro average) = 0.5754027527729523
F1 score (macro average) = 0.5778603605067937
              precision    recall  f1-score   support

           0       0.66      0.46      0.54      3972
           1       0.61      0.72      0.66      5937
           2       0.52      0.55      0.53      2375

    accuracy                           0.60     12284
   macro avg       0.59      0.58      0.58     12284
weighted avg       0.61      0.60      0.60     12284

[[1840 1780  352]
 [ 824 4251  862]
 [ 136  940 1299]]
```

Figure 6: Screenshot of the results.

Accuracy of 0.6016 indicates that the model correctly predicts the sentiment of approximately 60% of the tweets in the test set. While this is above chance level, it suggests there's room for improvement. The model is relatively good at predicting negative sentiment correctly when it does predict it (precision of 0.66), but it misses a significant number of negative instances (recall of 0.46). This results in a moderate F1-score of 0.54.The model performs best in predicting positive sentiment, with a relatively high recall of 0.72, indicating it captures most of the positive instances. The model struggles the most with neutral sentiment, having the lowest precision (0.52) and recall (0.55). This can be due imbalanced class distribution. This a bar plot given below for distribution of predicted and actual labels:

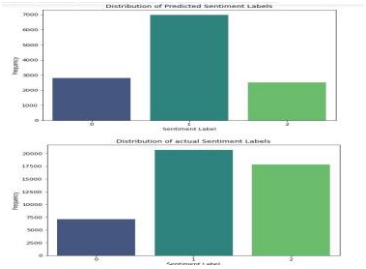

Figure 7: Screenshot of the results.

The use of bigrams in Model 2 did not lead to an improvement in performance. In fact, it resulted in lower accuracy, macro-averaged precision, recall, and F1 score compared to the unigram model (Model 1). Therefore, for this dataset, bigrams did not improve performance. Adding lexicon features also did not improve performance.This is because adding lexicon features increases the feature space. If these new features are not highly informative, they can introduce noise, making it harder for the model to learn useful patterns.The lexicon features could be redundant, leading to marginal or no improvement. This can slightly degrade performance rather than improve it. This proves the hypothesis that it will improve accuracy wrong.Our model performs better than Random Forest and support vector classifiers. This is evidenced by the accuracy scores as we receive an accurarcy score of 0.59 for Logistic Regression but 0.54 for Random forest and 0.58 for SVC classifier.We can improve our model through resampling techniques. Regularization techniques, such as L1 (Lasso) or L2 (Ridge) regularization, can help prevent overfitting by penalizing large coefficients, leading to a more generalizable model. It is also crucial to perform hyperparameter tuning to find the optimal regularization strength, which can be achieved through cross-validation methods[5]. Hyperparameter training could be performed using BayesSearchCV.

3 Named Entity Recognition (max 20%)

3.1

Named entity recognition (NER) is the task of identifying and classifying token-level instances of named entities (NEs), in the form of proper names and acronyms of persons, places or organizations, as well as dates and numeric expressions in text. First, we prepare data by Combining FIN5.txt and FIN3.txt datasets. Then we Ensure each sentence is tokenized and POS-tagged. For each sentence, we extract entities labeled in the categories: LOC, ORG, PER, and MISC. The Part-of-speech tags of the tokens are also extracted. Then we place all these information in a dataframe with columns sentence id, word, POS and tag. sentence _id contains the id of the sentence to which the word belongs, column word contains the word itself, column POS contains the part of speech the word belongs to like NN for noun and tag contains the entity type like I-PER. Then we transform the text data to vector and then split to train and test sets. We use Conditional Random fields(CRF) model. Then we define a SentenceGetter for preparing and organizing sentence data before training a CRF model. Each sentence is represented as a list of tuples where each tuple contains a word, its POS tag, and its NER tag.This structured format is suitable for training sequence labeling models like CRFs. Then we do feature extraction to like word parts, simplified POS tags, lower/title/upper flags, features of nearby words, and convert them to sklearn-crfsuite format — each sentence should be converted to a list of dicts. Then split the sentences into 33% test set and the rest as training set. Then we train the crf model using the training set and evaluate the model using the test set. CRFs are excellent at modeling the context in sequences, capturing the dependencies between neighboring labels. CRF allows the inclusion of various features which can significantly improve the model's performance. CRFs have shown high accuracy for sequence labeling tasks, making them suitable for NER. Like every model, CRF has some limitations as well. Training CRF models can be computationally expensive, especially with large datasets. Requires careful and often extensive feature engineering to achieve good performance. Our findings indicate that in-domain data is crucial, and mixing it with out-of-domain data can degrade performance. The Word Parts

feature Helps in identifying prefixes and suffixes which are often indicative of certain entity types. The Simplified POS Tags feature Provides syntactic context, helping the model understand the role of the word in a sentence. The Lower,Title and Upper Flags feature Identifies capitalization patterns which are important for recognizing proper nouns. The Features of Nearby Words captures the context around a word, helping to identify entities based on their surroundings. These features were chosen because they collectively capture both the individual characteristics of the words and their context within sentences, which is crucial for accurate named entity recognition. There are some challenges presented by the dataset. The financial domain has specific terminology and jargon not commonly found in general NER datasets. This requires domain-specific knowledge for effective tagging. Financial texts can contain ambiguous entities where the same term might be used for different types of entities based on the context. Certain entity types, particularly MISC, might be underrepresented in the dataset, making it challenging for the model to learn these categories effectively. Manual annotations can be inconsistent or incorrect, introducing noise into the training data which can affect the model's performance. Financial agreements often contain long and complex sentence structures which can be challenging for sequence labeling models to parse correctly. Misspelling, slangs, emoticons and abbreviation affect the quality of data and consequently the performance of the model[4].By addressing these challenges through careful data preparation, feature engineering, and model selection, we should develop a robust NER system for financial agreements.

3.2

Performance metrics are used in the evaluation of the model. Precision is used, which measures the ratio of correctly predicted positive observations to the total predicted positives. High precision indicates a low false positive rate. Recall is used, which measures the ratio of correctly predicted positive observations to all observations in the actual class. High recall indicates a low false negative rate. F1-Score is used, which is Harmonic mean of precision and recall, providing a single metric that balances both concerns. Confusion Matrix is used which Shows the actual vs. predicted classifications, providing insights into where the model is making errors.While high precision is good, it might come at the cost of recall, meaning some actual entities might be missed. High recall might reduce precision, meaning the model might label too many false positives.F1-Score Balances precision and recall but doesn't provide insight into specific types of errors. Accuracy score is used, which is the ratio of the number of correct predictions to the total number of predictions made by the model.Accuracy score is sensitive to class imbalance and Accuracy treats all misclassifications equally, without considering the potential costs associated with different types of errors. In many real-world scenarios, misclassifying certain classes may be more costly or critical than others. The model has high precision across most classes, particularly

	precision	recall	f1-score	support
I-LOC	0.93	0.56	0.70	174
I-MISC	1.00	0.67	0.80	3
I-ORG	0.79	0.73	0.76	195
I-PER	0.98	0.93	0.96	331
O	0.99	1.00	0.99	16931
accuracy			0.99	17634
macro avg	0.94	0.78	0.84	17634
weighted avg	0.99	0.99	0.99	17634

Figure 8: Screenshot of the results.

for "I-MISC" and "I-PER", suggesting that when it predicts these classes, it is usually correct.The recall for "I-LOC" is relatively low (0.56), indicating that the model misses a significant number of "I-LOC" instances. This might be due to the complexity of correctly identifying location entities in the text.There is a significant class imbalance, with the "O" class having the highest support (16,931 instances) compared to the named entities. This can affect the model's performance, as it might become biased towards predicting the majority class ("O").The high F1-scores for "I-PER" (0.96) and "O" (0.99) indicate that the model performs exceptionally well in identifying person entities and non-entities. Our model has an accuracy score of 99% on unseen test data, which means it's predictions are correct 99% of the time. A good visualization of the confusion matrix is below: We can improve the model by

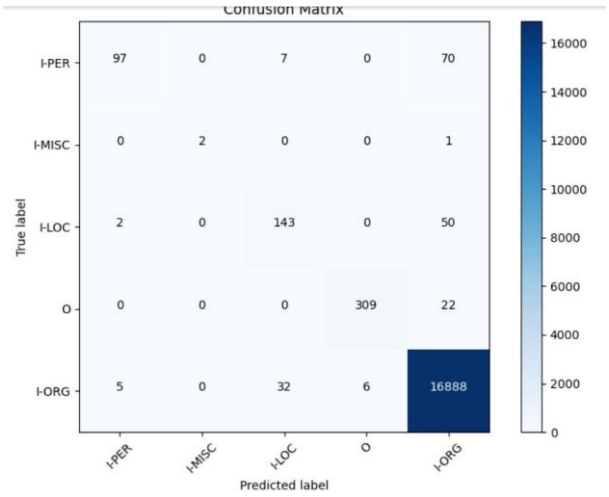

Figure 9: Screenshot of the results.

addressing Class Imbalance which involves using techniques like oversampling the minority classes or undersampling the majority class. Applying class weighting to give more importance to the minority classes during training. We should Incorporate additional features such as word embeddings (e.g., BERT, Word2Vec) to capture more semantic information. We should use contextual features like surrounding words, syntactic dependencies, and sentence structure. We should experiment with more advanced models like transformer-based models (e.g., BERT, RoBERTa) that can capture context more effectively. We should perform extensive hyperparameter tuning to find the optimal settings for the CRF model.

4 Information Visualisation (max 35%)

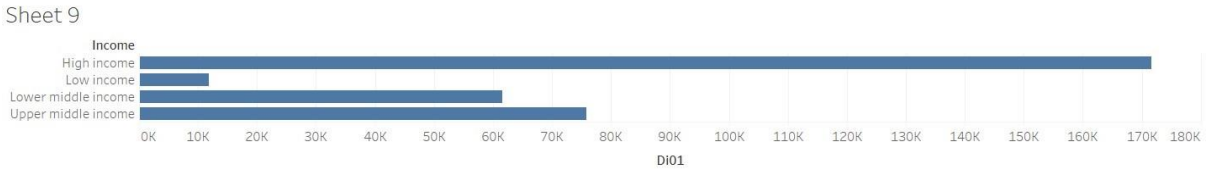

Figure 10: bar chart

The bar chart above displays the average private credit by deposit money banks to GDP (%) (GFDD.DI.01) across different income levels. Each bar represents an income level (e.g., low income, lower-middle income, upper-middle income, and high income), and its height corresponds to the average percentage of private credit by deposit money banks to GDP for countries within that income level. The bar chart is an effective way to compare discrete categories, such as income levels, allowing users to quickly identify differences in private credit to GDP ratios. Bar charts are particularly useful for comparing magnitudes because human perception is adept at judging the length of bars against a common baseline[3]. . The box plot displays the distribution of private credit by deposit money banks to GDP (%) for each income level. Each box represents the interquartile range (IQR), with the median marked inside the box. Clarity: Both the bar chart and box plot present the data clearly, with minimal extraneous details. This aligns with Edward Tufte's principles of clarity and simplicity in data visualization[2] .The bar chart enables straightforward comparison of average private credit by GDP across income levels, while the box plot

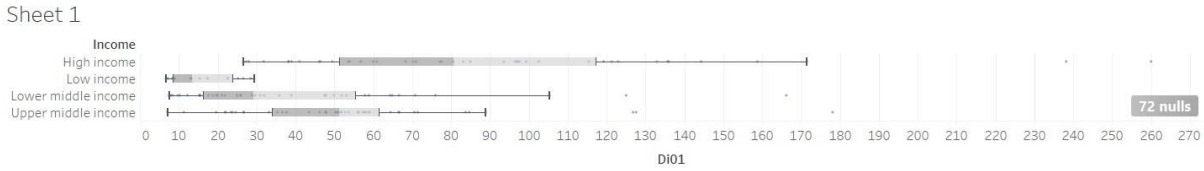

Figure 11: Screenshot of the box plot.

allows users to compare distributions. According to Few's guidelines on visual design, these techniques facilitate quick and accurate comparisons .Utilizing these visual forms leverages pre-attentive processing capabilities of the human brain, allowing users to grasp patterns and differences rapidly[3].These visualizations enable users to explore the relationship between private credit and income levels interactively. By providing both a summary (bar chart) and detailed distribution (box plot), users can derive insights at both the macro and micro levels[1]. In the second part, when we talk about significant rise and fall, we what essentially interests users is

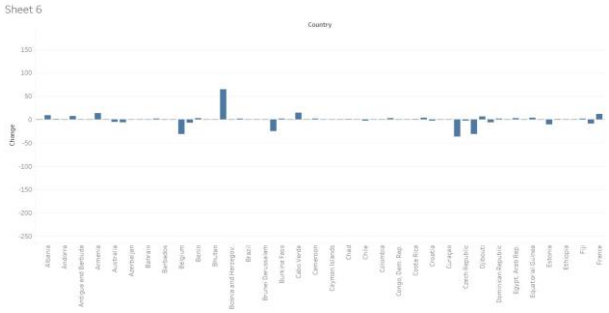

Figure 12: Screenshot of the bar chart.

a change. Change is positive for rise and negative for fall. So, I calculated a field for the change in value from 2004 to 2020. Then, I employed two visualization techniques in Tableau: bar charts and maps. These techniques were chosen based on principles of information visualization and considerations of human perception and cognition. In the bar chart, Positive values indicate a rise, while negative values denote a fall. Each bar's length reflects the magnitude of change, with longer bars indicating more significant changes. Using a bar chart with positive and negative values provides a straightforward visual representation of the direction and magnitude of changes in bank branches over time. Viewers can easily compare the extent of changes across different countries by observing the lengths and directions of the bars focusing solely on changes, the visualization avoids potential confusion associated with absolute values, making it easier for users to grasp the overall trends. It maps leverage spatial perception to provide users with a global context, allowing them to identify regions or countries with notable increases or decreases in bank branches. Color-coded symbology enables users to quickly recognize positive and negative changes, facilitating rapid comprehension of the overall distribution of changes across different geographic areas. By incorporating hover-over functionality to display precise percentage changes, users can interactively explore the data for each country, gaining insights into specific changes within their geographical context. The image below, i.e. figure 13 is found in sheet 12. Please zoom the image below to see it as we have page limitations.

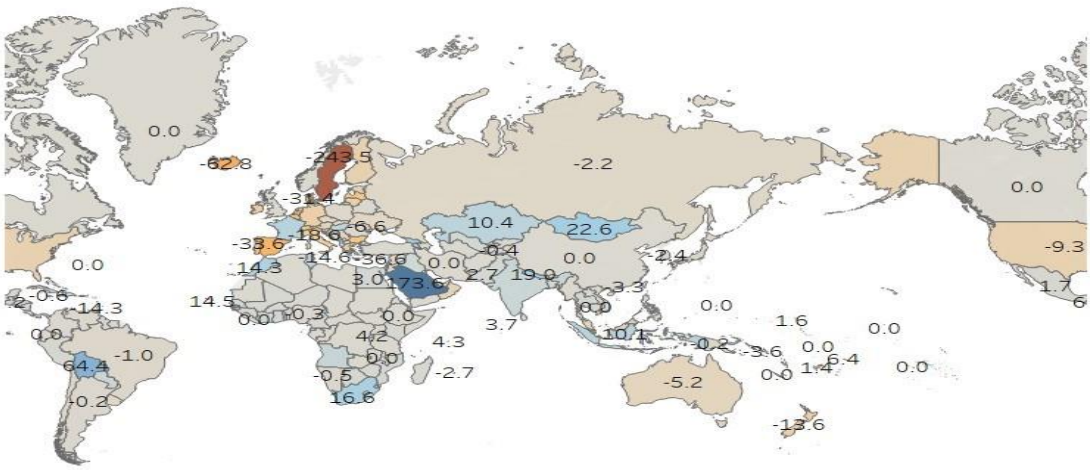

Figure 13: Screenshot of the results.

A world map was chosen to display the data geographically, allowing users to quickly grasp spatial patterns and regional differences. Different colors were used to represent varying levels of bank concentration (%). This helps in distinguishing high, medium, and low values across different countries. It is easy for the user to catch as more reddish means more concentrated. Interactive tool tips were added to provide information when the user hovers over them. This includes the name of country and the exact values for bank concentration and bank deposits to GDP. This makes it highly user interactive. Numbers were marked above each country to provide a clear, immediate reference to the specific values of banks deposit to gdp ratio. This also makes sure both indicators were layered on the same map, allowing users to compare them simultaneously and also if bank concentration is higher then bank deposits to gdp are likely to be higher, as demonstrated in the map. The map below is found in sheet 7 of my tableau workbook.

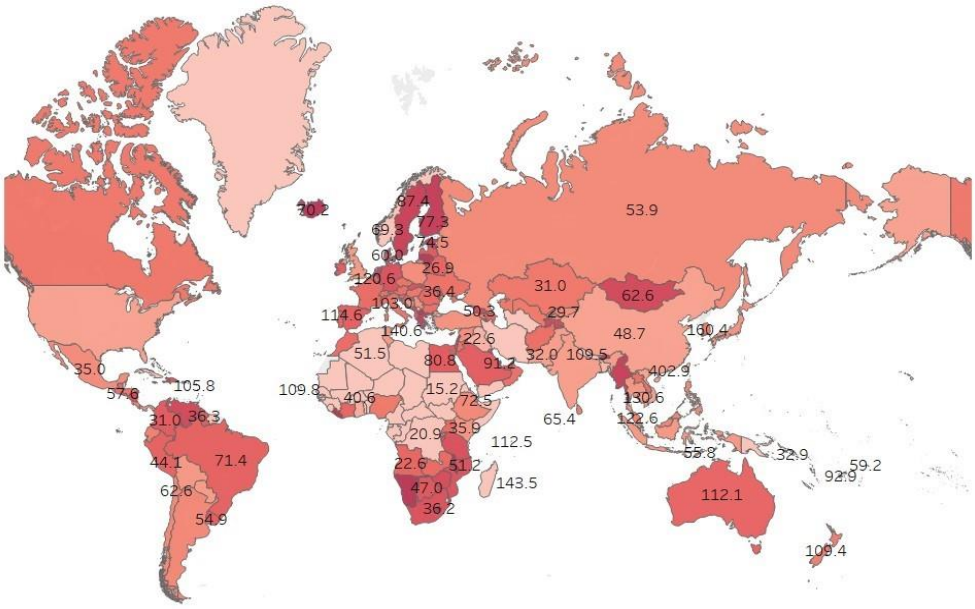

Figure 14: Screenshot of the results.

References

[1] William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.

[2] Edward R Tufte. The visual display of quantitative information. 1983.

[3] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[4] Kavitha, D., Venkatraman, S., Karthik, C.R. and Nair, N.S., 2024. Machine Learning-Based Sentiment Analysis of Twitter Using Logistic Regression. In *Advancing Software Engineering Through AI, Federated Learning, and Large Language Models* (pp. 308-319). IGI Global.

[5] Yang, L. and Shami, A., 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, pp.295-316.