The School of Mathematics

THE UNIVERSITY
of EDINBURGH

My Incredible Thesis

by

Robin Lin

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

Wednesday 28th June, 2023

Supervised by
Dr Nicole Augustin and Dr Michael Allerhand

Executive Summary

Here comes your executive summary ...

Acknowledgments

Here come your acknowledgments ...

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: ....
Matriculation Number: ....
Title of work: ....

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature ....

Date ....

Contents

List of Tables

List of Figures

1 Introduction

1.1 Overview

Simply Business is one of the largest providers of business and landlord insurance in the UK. Such a provider works on determining premiums that customers need to pay for their property insurance. When the premiums are being estimated, various risk factors are taken into account, including the types and sizes of properties, the number of rooms, and the characteristics of the constructions, etc. Among all risk factors, the types of properties are vitally important for insurance companies. A possible reason for that could be the variations in shapes and materials in the construction processes of properties. Such differences could have an impact on whether they are easy to get damaged in various situations. For instance, the ability of a round-shaped facade to avoid being knocked down by strong wind is often higher than that of a flat one. Also, brick houses are not as easily burnt by fire as wood ones. Such information is used by insurance providers to adjust the policies to their customers accordingly.

1.2 Data Description and Project Objectives

In this project, the street view images and datasets to be used are provided by *Simply Business*. Multiple images are extracted from the latest Google street view data, with a unique identity number attached to each of them. Multiple datasets are extracted from *Rightmove* website, clearly showing the necessary information of the properties as follows.

- Addresses
- Property Types
- Numbers of Bedrooms
- Web Links
- Location Coordinates
- Identity Numbers

For the images, their property types could be extracted from one of the given datasets by providing the identity numbers attached to them, allowing the types of properties to be linked with the images. The main objectives of this project are to estimate the types of properties by constructing different models given the images from Google street view, and to select a model with the best accuracy, so that the types of properties could be detected automatically. It is worth estimating the types of properties, in that some customers might have failed to reveal the information regarding their properties, or they might have provided the wrong information. Since images from Google street view data are novel, such intuitive and figurative information could be able to reveal the housing status of a customer authentically.

1.3 Characteristics of the Properties and Street View Images

As mentioned above, the datasets clearly show the information of the properties. These datasets also share exactly the same attributes, making it easy for us to merge the entries from different datasets into a new data frame. In the new data frame, there are $37,402$ entries in total, and no missing values are found for the property types. However, not every property has an attached street view image. In this case, only the properties with street view images attached would be taken into account in the models. The number of such kind of properties is $35,744$. However, this does not necessarily mean that the number of unique street view images is $35,744$, and in fact a large number of duplicate images are spotted. Although the duplicate images have different identity numbers attached to them, they share exactly the same latitudes and longitudes. In the future models, overfitting issue might occur if a large amount of duplicate images are taken into account in the models, since there would be more noise affecting the performances of the models. In view of this, such duplicate images should be discarded, and there are $15,484$ unique images. In the new data frame, the entries corresponding to the duplicate images should also be removed accordingly. Those entries are the ones with the same values of latitudes and longitudes. The number of unique properties is hence $15,484$.

The unique properties are scattered on the UK map as follows in Figure 1.

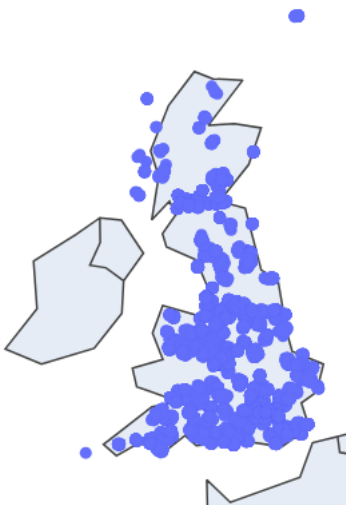

Figure 1: Locations of the Properties

At first glance, it could be observed that the properties are selected from Great Britain and some other small isles of Britain. By having a closer look into it, it could be observed that the properties are mostly extracted from the middle and south of England, north of Wales, Glasgow, and Edinburgh. There are also a few properties located in the north of England, as well as in the north of Scotland.

There are 4 types of properties considered in this project, which are detached, flat, semi-detached, and terraced. However, there are 5 classes in the variable "property types", and the other class not specified above is the "unknown" one. When street view images are labelled as "unknown", there could be at least one possible type of property, or it could be the case where none of the 4 types best describe the characteristics of the properties in the street view images. The percentages of the 5 classes are shown as follows in Figure 2.

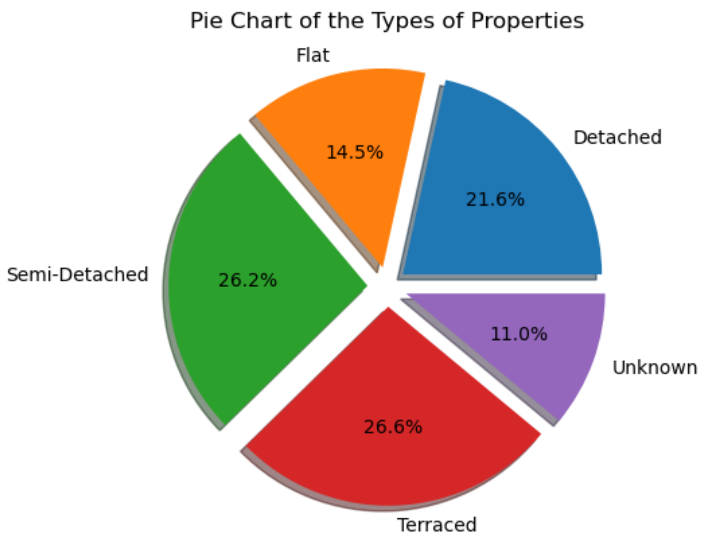

Figure 2: Pie Chart of the Types of Properties

From the pie chart, it could be observed that the classes are not evenly divided, indicating that the data set is imbalanced. An imbalanced dataset could be harmful in the models. When a model is being trained, it could probably keep predicting the classes that have higher proportions, in order that a higher accuracy in the training sets could be achieved. However, when it comes to validating and testing the model, the performances of classes with lower proportions might not be decent enough. In addition, classes with lower proportions might not be able to learn as much as the other classes do, leading to lower accuracy in model prediction. In view of this, some perturbations are to be made in the loss function, which would be discussed in later sections.

As mentioned above, there are 5 classes in the variable "property types", and one of the classes is "unknown". It is crucial to discuss whether to have the images labelled as "unknown" kept as they are, or to have them discarded, as this might have an impact on the performances of the models. There is no harm in looking at some example images first. As mentioned above, it could be the case where none of the 4 types best describes the characteristics of the properties. As shown in Figure 3, if images mostly contain objects that are in green, such as trees or forests, or when they contain a road without a house, the models might learn them as a pattern, and classify property types correctly as "unknown". However, such a case only occurs in a very small portion of images.

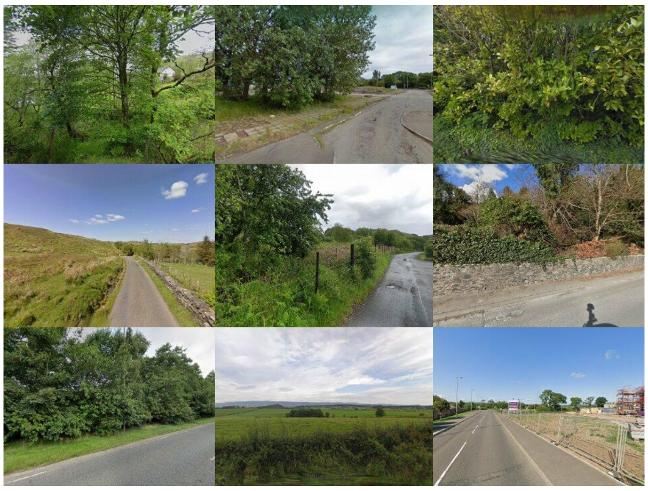

Figure 3: Examples of Properties Labelled "Unknown" without a House

In general, there are houses in most images labelled "unknown", so there is a greater chance that at least one possible type of property is in the images, as shown in Figure 4.

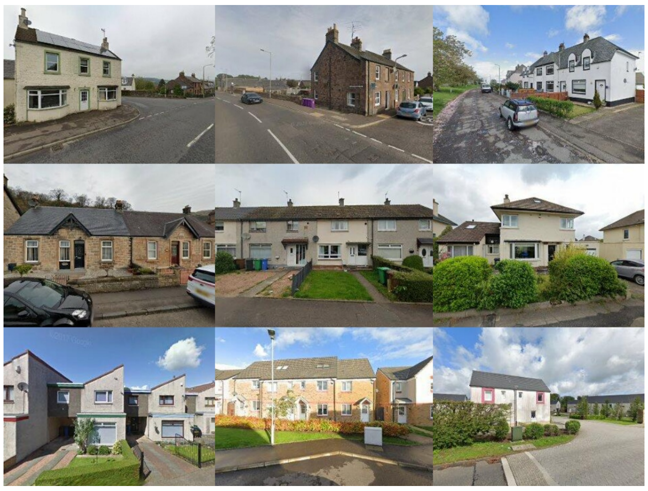

Figure 4: Examples of Properties Labelled "Unknown" with Houses

When models are being trained, the contours of the houses might be captured and sketched by them. After the models are trained, they should be validated and tested, and the contours are key information in classifying the types of houses. This could lead to a situation where the models classify objects in the images as one of the 4 types of properties, instead of classifying them as "unknown", indicating that the accuracy of validation and testing would be very low. Moreover, “unknown” is not a type of property being widely acknowledged by people, and hence it does not make any sense to classify the images as this type. Therefore, the images labelled “unknown” are ignored, along with their corresponding entries in the new data frame. The number of images to be considered should hence be $13,775$.

1.4 A Literature Review on Image Classification Models

A brief review of the literature is provided, presenting the models applied in classification of images, especially Google street view images, in recent years. In 2016, the authors of [1] extracted visual and textural features, and fed them into a multi-layer perceptron (MLP) model and a support vector regression (SVR) model. They concluded that their MLP model performed better than their SVR model. In 2017, the authors of [6] applied the convolutional neural network (CNN) model to classify the frontages of street view images in Greater London and found decent results in their model. In 2018, the authors of [4] applied VGG16, a special kind of CNN model, to classify the facade structures from Google street view images, and achieved high accuracy. In 2019, the authors of [5] applied CNN and Generalised Additive Model (GAM) to the Google street view images from different angles in predicting house prices in London, and found promising results. In 2020, the authors of [2] adopted Graph Neural Network (GNN) models in classifying superpixel images, which was an unprecedented experiment, and they successfully classified panoramas. Finally, in 2021, the authors of [7] also adopted VGG16 to detect visual cracks in pavement images, which were extracted from Google street view, and managed to classify those images into different categories of cracks.

From the above literature, it could be observed that people tend to use CNN, especially the newly invented VGG16, to classify street view images. There are cases where people use the traditional MLP and SVR models. In this project, more than $10,000$ images are to be trained, and it does not seem time-efficient to train such a large amount of images using SVR models. Moreover, VGG16 model looks too fancy to be implemented, since such kind of model has a complex structure, and overfitting issue might occur more easily when such a model is adopted. Therefore, simple MLP and CNN models are adopted in this paper.

2 Preliminary Settings

2.1 Downsizing Images

The street view images are reformatted into matrices with 3 axes. The first axis describes the height of an image, and there are 320 pixels along this axis. The second one describes the width of an image, and there are also 320 pixels along this axis. The third one describes the number of primary colours. After reformatting, $307,200$ pixel values are generated for each image, and such values are integers ranging from 0 to 255. However, there could be an issue of memory if a large amount of features are taken into account in constructing models. A possible way to avoid it could be applying dimensionality reduction techniques, including principle component analysis (PCA), and factor analysis. However, chances are that crucial information might be lost when applying such techniques. For instance, contours of the houses might be necessary for image classification, and if some of those key pixels are missing, models might provide ridiculous results, such as classifying a flat as a terraced house. In view of this, dimensionality reduction techniques would not be the best option. Another possible way could be extracting features by applying algorithms like autoencoder, or diffusion map, etc. However, none of these algorithms seem time-efficient in dealing with such a large scale of features in the images. The fastest, and perhaps the most straightforward method is resizing those images to a lower resolution. It should be very fast, since it is just about merging, and averaging neighbouring pixels. It should also preserve the key information in the images, since downsizing them would preserve the information of the contours of houses, which might be crucial in modelling. In this project, the heights and widths of the images are downsized to 64 pixels. The main advantages of doing so are that training a model would not be extremely time consuming, and that key information, such as the contours of houses, would be preserved.

2.2 One Hot Encoding

For types of properties, the categorical outcome variable, one hot encoding is applied to convert it into the form of numerical values. Such an encoding method preserves norminality of the variable, ensuring that the models learn patterns of the outcome variable.

2.3 Train-Validation-Test Split

The models adopted in this paper are neural network models. In order to avoid overfitting, it is necessary to split the dataset into different groups. In this project, the dataset is split into non-overlapping training, validation, and testing sets by randomly selecting an integer-valued seed. Among the 3 sets, there are 70% of the data in the training set, 20% in the validation set, and 10% in the testing set.

3 Modelling

3.1 Normalising Pixel Values

As mentioned above, the pixel values range from 0 to 255. The first thing that we should think about is to normalise these values, so that they range from 0 to 1. There are some good reasons for this regarding how the models update the weights. For neural network models, the weights in a certain iteration are updated by adding the product of the learning rate and the gradient to the weights in the previous iteration. If the pixel values were not scaled, the corrections that they might have produced would vary from one another. In other words, some weights might be over-corrected, while some others might be under-corrected. In order to make similar corrections to the weights, such pixel values should be normalised. Typically, in this case, the normalisation process could simply be done by dividing the pixel values by 255.

3.2 Selecting Activation Functions

3.2.1 ReLU or Leaky ReLU?

ReLU stands for rectified linear unit. Such an activation function returns exactly the input when it is non-negative, and returns 0 when it is negative. Compared with sigmoid and hyperbolic tangent activation functions, it does not suffer from the problem of vanishing gradient. When one of sigmoid and hyperbolic tangent activation functions is applied, the gradient has an upper bound, and the further the inputs are from 0, the smaller the gradient would be. This indicates that the models learn the patterns more slowly when there are multiple layers, and hence the weights would barely update. In order that the models could be persistently learn patterns in the images, ReLU is a better choice compared to sigmoid and hyperbolic tangent activation functions.

Leaky ReLU stands for leaky rectified linear unit. It returns exactly the same value as it is in ReLU when the input is non-negative, but for negative inputs, it has a small slope. Such a small slope is almost invisible, but it is super powerful that it achieves what a ReLU function would not be able to accomplish. When the inputs are negative, ReLU function keeps outputting 0 all the time, making the neurons inactive. In this case, models would no longer learn anything. However, the leaky ReLU function outputs a non-zero value when the inputs are negative, triggering the neurons to learn the patterns, so that the weights could be updated. This is useful, since the street view images might contain noise, and adopting such an activation function would make models more robust.

By and large, the leaky ReLU activation function is applied in all of the layers that need an activation function, except for the output layer.

3.2.2 Why Softmax in the Output Layer?

In this project, there are 4 types of properties, as mentioned above. This is a multi-class classification problem, so the activation functions for binary class classification are inappropriate in the output layer. In this case, a typical activation function placed in the output layer is the *softmax* function.

In a sentence, *softmax* function converts the inputs into different probabilities that sum up to 1. In neural network models, these probabilities refer to the likelihoods of predicting different classes.

3.3 Choosing an Optimisation Algorithm

The optimisation algorithm we have chosen for the models is Adam. This is by far the best optimisation algorithm in constructing neural network models, and it has several advantages. Firstly, it does not require a large amount of memory, and it is simple and effective in its implementation. It is also capable of problems having a large amount of data, especially for such a massive image classification problem containing $13,775$ street view images. Finally, the hyper-parameters are rather easy to tune and interpret, making it easy to design Adam for different deep learning problems.

3.4 Monitoring the Models

3.4.1 Epochs

In neural network models, an epoch refers to a complete training process for all data in the training set. The more the epochs are, the more the chances for the models to learn the patterns in the training set. However, it is also possible that the overfitting issue might occur. In order that this issue could be avoided, the models should be terminated from training when the validation accuracy is not getting any higher for several epochs. In this project, the models terminate from training if the validation accuracy is not improving after 5 epochs in a row. This saves the training time considerably, and prevents the models from learning too much noise.

3.4.2 Learning Rates

The default learning rate of the Adam algorithm is 10^{-3}. However, in the models, the learning rate should not be a constant, since there is neither guarantee that the problem is a convex one, nor guarantee that the validation accuracy is bound to converge. In our case, the learning rates should be adjusted accordingly based on the performances on the validation accuracy in each epoch. The initial value of the learning rate is set to be half of the default value, *i.e.*, 5×10^{-4}, since we do not wish the models to learn greedily at the very beginning, avoiding the possible low validation accuracy. Every time the accuracy decreases, or when it fails to reach the highest value in previous epochs, the learning rate is multiplied by 4×10^{-2}. Usually, the models would learn the patterns effectively when the learning rate ranges from around 10^{-6} to around 10^{-3} within the first 7 or 8 epochs, and after that, the validation accuracy starts to converge.

3.5 Choosing Batch Sizes

In neural network models, the batch size refers to the number of samples used to update the weights. The larger the batch size, the smaller the number of iterations in an epoch. In practical experiments, the larger the batch size, the shorter the training time in an epoch, and the larger the variation in the validation accuracy. The smaller the batch size, the longer the training time in an epoch, and the higher the risk of encountering the issue of overfitting. In order to achieve a high and robust validation accuracy, a trade-off between them should be taken into account. In the models, the batch size is selected to be 50, ensuring that the training time is not that long, and the validation accuracy grows steadily.

3.6 Assigning Weights to the Loss Function

3.7 Regularisation

3.8 Models

3.8.1 MLP Model

3.8.2 CNN Model

4 Results

I this section, I explain what did I discover.

Now it's getting very technical ... I will cite [?, ?]. I will also show my incredible α, β and γ mathematics and do some other fancy stuff.

4.1 Formulae

For example look at this

$$\min \sum_{s\in\mathcal{S}} Pr_s \left[\sum_{t=1}^{T}\left(\sum_{g\in\mathcal{G}}\left(\alpha_{gts}C_g^0 + p_{gts}C_g^1 + \left(p_{gts}\right)^2 C_g^2\right) + \sum_{g\in\mathcal{C}} \gamma_{gts}C_g^s\right)\right], \tag{4.1}$$

and you will see that it has a little number on the side so that I can refer to it as equation (4.1). Now if I do this

$$\begin{aligned} \sum_{i=1}^{n} k_i &= 20 \qquad (4.2) \\ \sum_{j=20}^{m} \delta_i &\geq \eta \end{aligned}$$

I can align two formulae and control which one has a number on the side. It is (4.2). I can also do something like this

$$Y_l = \begin{bmatrix} \left(y_s + i\frac{b_c}{2}\right)\frac{1}{\tau^2} & -y_s\frac{1}{\tau e^{-i\theta^s}} \\ -y_s\frac{1}{\tau e^{i\theta^s}} & y_s + i\frac{b_c}{2} \end{bmatrix},$$

and it won't have a number on the side. Now if I have to do some huge mathematics I'd better structure it a little and include linebreaks etc. so that it fits on one page.

$$\begin{aligned} p_l^f &= G_{l11}\left(2v_{F(l)}\bar{v}_{F(l)} - \bar{v}_{F(l)}^2\right) \qquad (4.3) \\ &+ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ &+ \begin{bmatrix} \bar{v}_{T(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[B_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) - G_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[-B_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \end{bmatrix} \cdot \begin{bmatrix} v_{F(l)} - \bar{v}_{F(l)} \\ v_{T(l)} - \bar{v}_{T(l)} \\ \delta_{F(l)} - \bar{\delta}_{F(l)} \\ \delta_{T(l)} - \bar{\delta}_{T(l)} \end{bmatrix}, \end{aligned}$$

This is a lot of fun!

4.2 Important Things

Finally we should have a nice picture like this one. However, I won't forget that figures and table are environments which float around in my document. So LaTeX will place them wherever it thinks they fit well with the surrounding text. I can try to change that with a float specifier, e.g. [!ht]. classes are shown as follows in Figure 2. Figure 2: Pie Chart of the Types of Properties From the pie chart, it could be observed that the classes are not evenly divided, indicating that the data set is imbalanced. An imbalanced dataset could be harmful in the models. When a model is being trained, it could probably keep predicting the classes that have higher proportions, in order that a higher accuracy in the training sets could be achieved. However, when it comes to validating and testing the model, the performances of classes with lower proportions might not be decent enough. In addition, classes with lower proportions might not be able to learn as much as the other classes do, leading to lower accuracy in model prediction. In view of this, some perturbations are to be made in the loss function, which would be discussed in later sections. As mentioned above, there are 5 classes in the variable "property types", and one of the classes is "unknown". It is crucial to discuss whether to have the images labelled as "unknown" kept as they are, or to have them discarded, as this might have an impact on the performances of the models. There is no harm in looking at some example images first. As mentioned above, it could be the case where none of the 4 types best describes the characteristics of the properties. As shown in Figure 3, if images mostly contain objects that are in green, such as trees or forests, or when they contain a road without a house, the models might learn them as a pattern, and classify property types correctly as "unknown". However, such a case only occurs in a very small portion of images. Figure 3: Examples of Properties Labelled "Unknown" without a House In general, there are houses in most images labelled "unknown", so there is a greater chance that at least one possible type of property is in the images, as shown in Figure 4. Figure 4: Examples of Properties Labelled "Unknown" with Houses When models are being trained, the contours of the houses might be captured and sketched by them. After the models are trained, they should be validated and tested, and the contours are key information in classifying the types of houses. This could lead to a situation where the models classify objects in the images as one of the 4 types of properties, instead of classifying them as "unknown", 3

the document with the pdflatex compiler.

Now I want to use one of my own environments. I want to define something.

Definition 4.1 *I define*

$$\Gamma_\eta := \sum_{i=1}^{n} \sum_{j=i}^{n} \xi(i,j)$$

I definitely need some good tables, so I do this. I should really refer to Table 1.

Case	Generators	Therm. Units	Lines	Peak load: [MW]	[MVar]
6 bus	3 at 3 buses	2	11	210	210
9 bus	3 at 3 buses	3	9	315	115
24 bus	33 at 11 buses	26	38	2850	580
30 bus	6 at 6 buses	5	41	189.2	107.2
39 bus	10 at 10 buses	7	46	6254.2	1387.1
57 bus	7 at 7 buses	7	80	1250.8	336.4

Table 1: Something that doesn't make sense.

4.3 And now something else

Let:

$$\begin{aligned}
\Omega_0 &= \{(x,y,z,f): \text{ satisfying } (9)-(19)\},\\
\Omega_1 &= \{(x,y,z,f): \text{ satisfying } (9),(11)-(20)\},\\
\overline{\Omega}_0 &= \{\mathbf{0} \le (x,y,z,f) \le \mathbf{1}: \text{ satisfying } (9)-(18)\},\\
\overline{\Omega}_1 &= \{\mathbf{0} \le (x,y,z,f) \le \mathbf{1}: \text{ satisfying } (9),(11)-(18),(20)\}\,.
\end{aligned}$$

where **0** and **1** are vectors of appropriate dimensions with 0's and 1's, respectively. Next we see that both Ω_0 and Ω_1 give equivalent formulations for the A-MSSP. In particular, the following statements hold:

Proposition 1 *$\Omega_0 \subseteq \Omega_1$.*

Proof. Let us suppose there exists $(x,y,z,f) \in \Omega_1$ such that $(x,y,z,f) \notin \Omega_0$. Then, there exist indices $i \in I$ and $t \in \{0,\ldots,|T|-s_i\}$ with $x_i^t > 0.5\left(\sum_{h=1}^{s_i} x_i^{t+h} + 1\right)$. By definition, $x_i^t = 1$ and $x_i^{t+h} = 0$ for all $h \in \{1,\ldots,s_i\}$. By (11) and (12), $\sum_{h=1}^{s_i} f_i^{th} = 1$, so $f_i^{th'} = 1$ for some $h' \in \{1,\ldots,s_i\}$. But then,

$$0 \;=\; x_i^{t+h'} \;=\; \sum_{h=\max\{1,t+h'-(|T|-s_i)\}}^{\min\{s_i,t+h'\}} f_i^{t+h'-h,h} \;\ge\; f_i^{th'} \;=\; 1\,,$$

as $h' \in [\max\{1, t+h'-(|T|-s_i)\}, \min\{s_i, t+h'\}]$. □

This immediately gives us

Corollary 1 *AS is a valid formulation for the A-MSSP.*

Next we compare the Linear Programming (LP) relaxations of the two formulations.

Proposition 2 *$\overline{\Omega}_1 \subseteq \overline{\Omega}_0$.*

Proof. Homework □

5 Conclusions

I have no idea how to conclude, so I don't write much. But the stuff that follows is important.

References

[1] E. Ahmed and M. Moustafa. House price estimation from visual and textual features. *arXiv preprint arXiv:1609.08399*, 2016.

[2] P. H. Avelar, A. R. Tavares, T. L. da Silveira, C. R. Jung, and L. C. Lamb. Superpixel image classification with graph attention networks. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 203–209. IEEE, 2020.

[3] F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[4] J. Kang, M. Körner, Y. Wang, H. Taubenböck, and X. X. Zhu. Building instance classification using street view images. *ISPRS journal of photogrammetry and remote sensing*, 145:44–59, 2018.

[5] S. Law, B. Paige, and C. Russell. Take a look around: using street view and satellite images to estimate house prices. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(5):1–19, 2019.

[6] S. Law, Y. Shen, and C. Seresinhe. An application of convolutional neural network in street image classification: The case study of london. In *Proceedings of the 1st Workshop on Artificial Intelligence and Deep Learning for Geographic Knowledge Discovery*, pages 5–9, 2017.

[7] M. Maniat, C. V. Camp, and A. R. Kashani. Deep learning-based visual crack detection using google street view images. *Neural computing and applications*, 33(21):14565–14582, 2021.

Appendices

A An Appendix

Some stuff.

```r
require(survival)
require(survminer)

# Kaplan-Meier plot by aetiology
km.aet <- survfit(Surv(TimeB, Status == 1) ~ Princ.aetio.simp,
data = screen.hcc.agg)

ggsurvplot(km.aet,
censor = F,
size = 0.5,
ggtheme = theme_gray(),
font.main = 9, font.x = 9, font.y = 9, font.legend = 8,
legend = "right",
xlab = "Time (days)") +
scale_color_hue(labels = c("ALD", "AID", "HepB", "HepC",
"NAFLD", "Other")) +
guides(color = guide_legend(title = "Princ. aetiology"))

# Log-rank test
survdiff(Surv(TimeB, Status == 1) ~ Princ.aetio.simp,
data = screen.hcc.agg)
```

B Another Appendix

Some other stuff.