Airbnb in London - Price Prediction

By Hui Zheng, Emre Dogan, Eamonn Kearney

Introduction

The Problem

We have chosen to work on the dataset of Airbnb, an online marketplace for travellers to find suitable accommodations from property owners around the world wanting to lease their premises on a short-term basis. We are interested to find out how machine learning can help in analysing the pricing of listings on Airbnb as well as to explore a machine learning model to predict prices.

In particular, this project attempts to answer the following **research questions:**

1. How are Airbnb prices in London distributed by borough?
2. How are Airbnb prices in London distributed by room type?
3. For listings that did not get any bookings, what does their price distribution look like?
4. Could pricing be the reason for the above listings not getting any booking?
5. What is a good machine learning model for predicting the price of an Airbnb listing?

We are focusing on pricing because it is a key factor for the success of the Airbnb business. It is also impacted by many variables, and we wanted to attempt to use machine learning to predict them.

Our dataset is obtained from InsideAirbnb (InsideAirbnb, no date), which has compiled information about Airbnb listings and data like calendar and other details. We used their dataset scraped on the 6th of November 2020 as this was the earliest dataset available on their website without having to pay for it. Datasets earlier than this had been archived and are available only if we paid for it.

Expected outputs of the analysis are:

1. Prices of listings in the inner London boroughs are expected to be more expensive than those of outer London, as the former properties are more centrally located and more expensive.
2. Prices of listings of “entire home/apartment” and “hotel room” are expected to be more expensive than those listing “private room” or “shared room”
3. Listings that did not get any bookings are expected to be deemed as “outliers”.
4. “Inappropriate pricing” is expected to be the main reason for listings not receiving any bookings.
5. As price prediction is a regression problem, Extreme Gradient Boosting (XGBoost) is expected to be our best-performing model as it combines several optimization techniques.

The Dataset:

The dataset consists of 76984 listings with the following characteristics:

- 74 columns describing different attributes
- Each listing’s data on price, borough, room types (this will help us find the price distribution across the various boroughs and the various room types.)
- Dates when each listing was *first* reviewed and *last* reviewed. Since there is no data on bookings, we assume that guests complete a review immediately after each stay. This means the number of reviews equals the number of bookings. This will help us find out about the listings that did not get any bookings.

The 74 features consist of a mixture of data types that are both discrete (e.g. bedrooms) and continuous (e.g. price). Other data types also include Booleans, arrays, date types, URLs, currencies, percentages, freeform texts in the types of categorical features.

The dataset is noisy and large, with many features that are non-numeric and with missing values. Time and scope limitations did not permit us to do any natural language processing (NLP), so the analysis excluded features with free text (e.g. description).

The steps below were taken to prepare the data:

- Identified non-numeric data types to convert them to numeric values:
  a. Used ones and zeros to replace features with Boolean values.
  b. Used One-Hot Encoding to encode features with categorical values.
  c. Converted special characters like "%" to float

- Missing values were removed, if more than 95% of the values were missing, or replaced with the mean or median.

- Outliers were identified and excluded from the analysis. For instance, about 2.1% of listings' prices were more than four standard deviations from the mean and were removed.

- Scaling/normalisation: Used MinMaxScaler as its performance was 0.1% better than StandardScaler.

- Used the following to perform multicollinearity analysis:
  a. Spearman's and Pearson's correlation
  b. Principal Component Analysis (PCA)
  c. Variance Inflation Factor (VIF)
  d. Manually

- Performed features selection based on the multicollinearity analysis.
  a. We inputted our data without removing any collinear features.
  b. We inputted our data using features that are greater than 0.10 in their correlation with price.
  c. We inputted our data with features from Spearman's and Pearson's correlation.
  d. We inputted our data without features with high VIF.
  e. We inputted our data without features that we had manually checked based on the correlation heatmap.

Many reiterations of the above were performed.

Data derivation:

- Derived "days_since_first_review" and "days_since_last_review" to calculate the number of days between the date of a listing's first/last reviews and the date that the dataset was scrapped.

  Features were created by converting first_review" and "last_review" features into DateTime objects, then subtracting them from "last_scraped". They were then categorised into periods of time and one-hot encoded.

- The dataset stored "amenities" as a single big block. As it was felt that this was a potentially rich data that could be used to answer Research Question 5 on price prediction, the amenities were split and separated by commas and then one-hot encoded.

Modelling:

Phase One – Model Selection

We used Linear Regression (LR) as the baseline as it has the advantage of being simple to set up and widely used in regression analysis. Its performance was compared with the following models:

- Gradient Boosting Regression (GBR)
- XGBoost
- Random Forest Regressor (RFR)
- Neural Networks (NN)

We split our dataset into training and testing sets using the **train_test_split** function from the Sklearn library. It makes random partitions for the two subsets. The training set is used to train, whilst the testing set is set aside to test the model's accuracy in the prediction. We started with a 70/30 ratio but found the result was better with a 80/20 ratio. Thus we split our dataset into 80% for training and 20% for testing.

The R-score (R^2), Root Mean Square Error (RMSE) and the **Mean Absolute Error (MAE)** were then used to evaluate each model's performance. RMSE and MAE are used to measure the accuracy of the prediction and how much it deviates from the actual price. Therefore we would want them to be as low as possible. Whereas R^2 is the proportion of variation in our prediction, thus we would want it to be as high as possible.

The first three models, GBR, XGBoost and RFR were chosen as they are popular ensemble models which combine several models in an ensemble to make better predictions. Random Forest uses the bagging technique whereas GBR and XGBoost use boosting technique. We decided to add Neural Network to our list as our dataset is large enough to train it and it is good at minimizing the loss function with backpropagation.

For the initial comparison, for NN, we used a Keras sequential model with 2 dense hidden layers (128 and 64 nodes respectively) sigmoid activation function, adam optimiser, and the number of epochs was set at 20. The other models were trained using their default parameters. The results of our initial comparison are shown in Table 1.

Table 1. Results of the initial model comparison

	Linear Regression	GBR	XGBoost	RFR	NN
R^2 (test)	0.4710	0.5211	**0.5898**	0.5841	0.5472
R^2 (train)	0.4758	0.5502	0.7844	0.9132	0.5365
RMSE (test)	68.900	65.292	60.427	61.678	**0.1187**
RMSE (train)	68.648	63.706	44.107	27.807	0.1194
MAE (test)	38.470	34.676	31.796	32.489	**0.0775**
MAE (train)	37.977	33.814	25.588	17.001	0.0780

Table 1 shows that the Linear Regression model generalised well but has the highest MAE and lowest R^2. Whereas the Random Forest Regressor model had the best R^2 for the training dataset but did not do as well in the testing dataset. This indicated that there is an overfitting issue. Although XGBoost had a better R^2 than NN, its RMSE and MAE results are way higher than NN.

In the next phase, we will deep dive into the models and try to improve their performance. Based on the above results and page count limitation, for the remainder of this report, we will focus on our training of Neural Networks and XGBoost.

Phase Two – Parameter Tuning

XGBoost is an implementation of the gradient-boosted tree that combines weak learners into a strong learner. It controls model complexity and has a built-in regularization to reduce overfitting (Quinto, 2020). It also has a built-in feature importance function that tells us which feature weighs more in the prediction of price. However, XGBoost does not perform well if the training set is very small or data is sparse and unstructured. It is not ideal for image recognition or natural language processing (Datacamp, no date). Since we did not intend to perform image recognition or NLP, we expected XGBoost to perform better from the tuning.

We improved the performance of XGBoost by tuning its hyperparameters. We optimized it by performing a GridSearchCV with cross-validation set to 3 and a RandomizedSearchCV with CV = 10. The parameters that performed the best were the ones returned by the GridSearch. There are as follow:

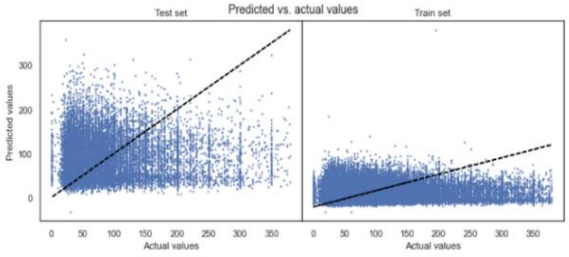
- 'colsample_bytree': 0.6
- 'gamma': 0.0
- 'learning_rate': 0.1
- 'max_depth': 7
- 'n_estimators': 200

The challenging part of this phase is the computational power it takes to run the GridSearch. As we were using brute force to find out the best parameters, it took many hours for the computer to return the results. We ran the trained model after each round of data cleaning and Table 2 displays the results of the final round and Fig. 1 shows the graph for parameters from GridSearch. The tuning of the parameters had decreased the MAE by 20.2% and increased the R^2 by 11.9%.

Table 2. Results of XGBoost in the final round

	XGBoost using corr() > 0.10	XGBoost using corr() > 0.10 and VIF	XGBoost after multicollinearity analysis	XGBoost using the parameters from GridSearchCV	XGBoost using the parameters from RandomizedSearchCV
R^2 (test)	0.6067	0.5921	0.6499	**0.6600**	0.6437
R^2 (train)	0.6866	0.6688	0.7500	0.7570	0.6998
RMSE (test)	41.588	42.352	39.236	**38.677**	39.582
RMSE (train)	37.200	38.239	33.222	32.752	36.407
MAE (test)	27.353	28.050	25.896	**25.387**	26.088
MAE (train)	24.907	25.823	22.223	21.702	24.020

Fig. 1. Graph of XGBoost with GridSearch



We wanted to know if the results will improve if we trained the XGBoost model with principal components from PCA. However, like the RFR, overfitting is an issue and the results are not better. It seemed like the Airbnb dataset does not reduce well in dimension. This means that most of the features are important as shown in Table 3 where the results improved when we increased the number of principal components.

Table 3. Results using PCA with XGBoost

	PCA = 2	PCA =30	PCA =60	PCA =100	PCA =110	PCA =120
R^2 (test)	0.0976	0.4670	0.4963	0.5692	0.5831	**0.5877**
R^2 (train)	0.1987	0.6866	0.7387	0.8073	0.8004	0.8041
RMSE (test)	62.614	48.115	46.777	43.260	42.556	**42.321**
RMSE (train)	59.592	37.269	34.029	29.220	29.740	29.468
MAE (test)	45.922	32.775	31.637	28.847	28.200	**27.957**
MAE (train)	43.656	26.002	23.839	20.332	20.689	20.415

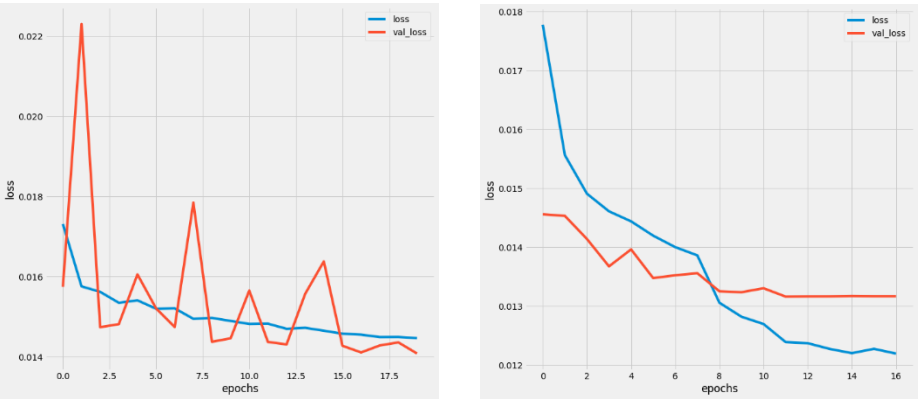
Neural Networks are good at learning, analysing, and finding the structure of input patterns (Lewis-Beck, Bryman and Futing Liao, 2004). They are also able to fit complex nonlinear models without the models being specified in advance. However, overfitting and overtraining could be an issue with NN. These issues can be resolved by selecting an appropriate architecture (Livingstone, Manallack, and Tetko, 1997). That is what we tried to do.

We optimised the performance of our NN incrementally across seven networks by first increasing the epoch count from 20 to 50, then increasing the number of units in the dense hidden layers from 128 and 64, to 1024 and 128, switching the sigmoid activation function to the relu activation function, adding a dropout layer with a dropout probability of 0.3. and introducing early stopping with patience 10 and later 5. Table 4 shows that the introduction of early stopping was able to reduce overfitting and Fig. 2 plots the loss.

Table 4. Results of optimising Neural Networks

	NN-1 Baseline	NN-2	NN-3	NN-4	NN-5	NN-6	NN-7
	Two dense hidden layers	Increasing epochs from 20 to 50	Increasing nodes from 128 and 64 to 1024 and 128	Switching to relu activation function	Adding early stopping with patience 10	Adding dropout layer, prob. 0.3	Decreasing early stopping patience to 5
R^2 (test)	0.5472	0.5512	0.5437	0.5165	0.5751	0.5751	**0.5767**
R^2 (train)	0.5365	0.5432	0.5373	0.7245	0.6201	0.6236	0.6172
RMSE (test)	0.1187	0.1181	0.1191	0.1226	0.1150	0.1150	**0.1147**
RMSE (train)	0.1194	0.1185	0.1192	0.0920	0.1081	0.1076	0.1085
MAE (test)	0.0775	**0.0758**	0.0760	0.0792	0.0760	0.0760	0.0759
MAE (train)	0.0780	0.0762	0.0762	0.0611	0.0723	0.0721	0.0728

Fig. 2. Final Neural Network loss



The baseline network (**left**) had a very unstable, relatively higher validation loss. Increasing the number of epochs increased the run time for the network, allowing more training. Changing the activation function in NN-4 was experimental but initially resulted in a validation loss actually rising while training loss plummeted, indicating severe inaccuracy and overfitting.

Early stopping in NN-5 helped reduce overfitting so that the model did not become too overtrained to the data, ideally increasing generalisation. A dropout layer further optimised the network in NN-6, bringing the losses closer together. The early stopping patience was reduced from 10 to 5 epochs to further minimise overfitting. The final result of NN-7 shows a relatively low loss with only a small degree of overfitting. The figures show loss and validation loss against epochs to demonstrate this. The difference in loss is relatively small, with the final training loss being around 0.012, and the validation loss being around 0.013 (**right**). This could be further optimised by several measures, like including the use of a convolutional neural network and further configuration experimentation.

Findings:

Research Question 1: How are Airbnb prices in London distributed by borough?

Fig. 3. London's Airbnb price distribution by borough

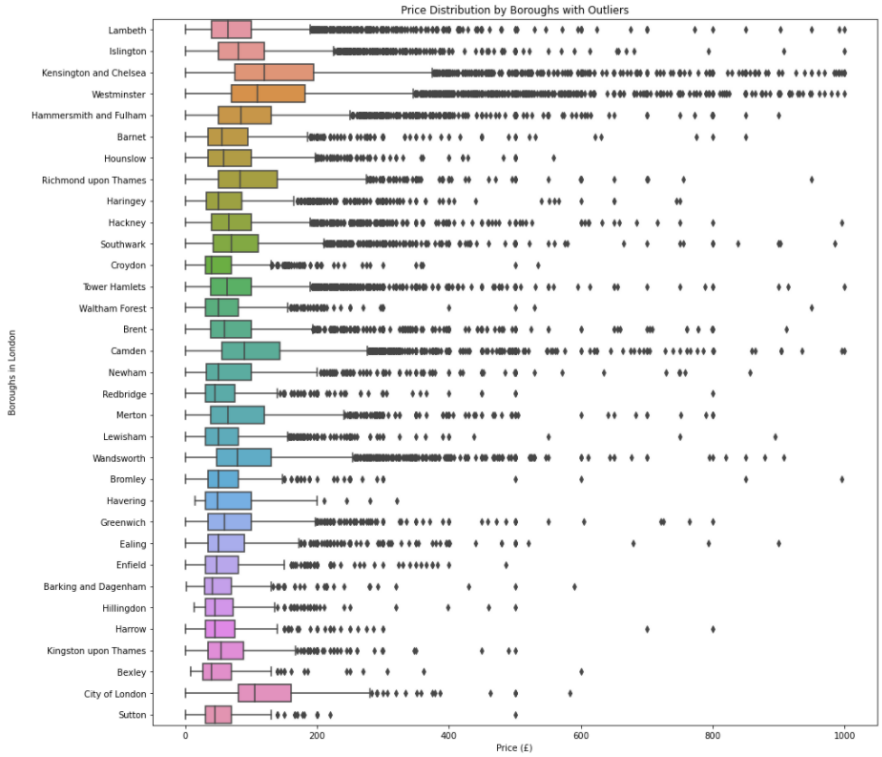


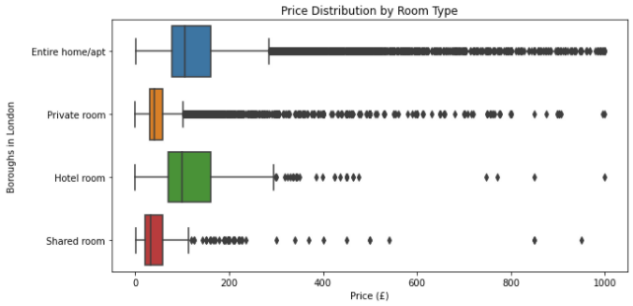
Fig 3 shows that the price distribution in London by borough is as we had expected, with prices in the inner London boroughs higher than those of outer London.

49.4% of inner London listings were priced at four times the standard deviation, while 28.88% and 20.52% of those in "Westminster" and "Kensington and Chelsea" are priced at four times the standard deviation respectively.

Research Question 2: How are Airbnb prices in London distributed by room type?

As we had expected, Fig. 4 shows that Hotel Rooms and Entire Homes are the most expensive on average, followed by Private Rooms and Shared Rooms. Entire Homes contain 86.36% of the outliers in price.

Fig. 4. London's Airbnb price distribution by room type

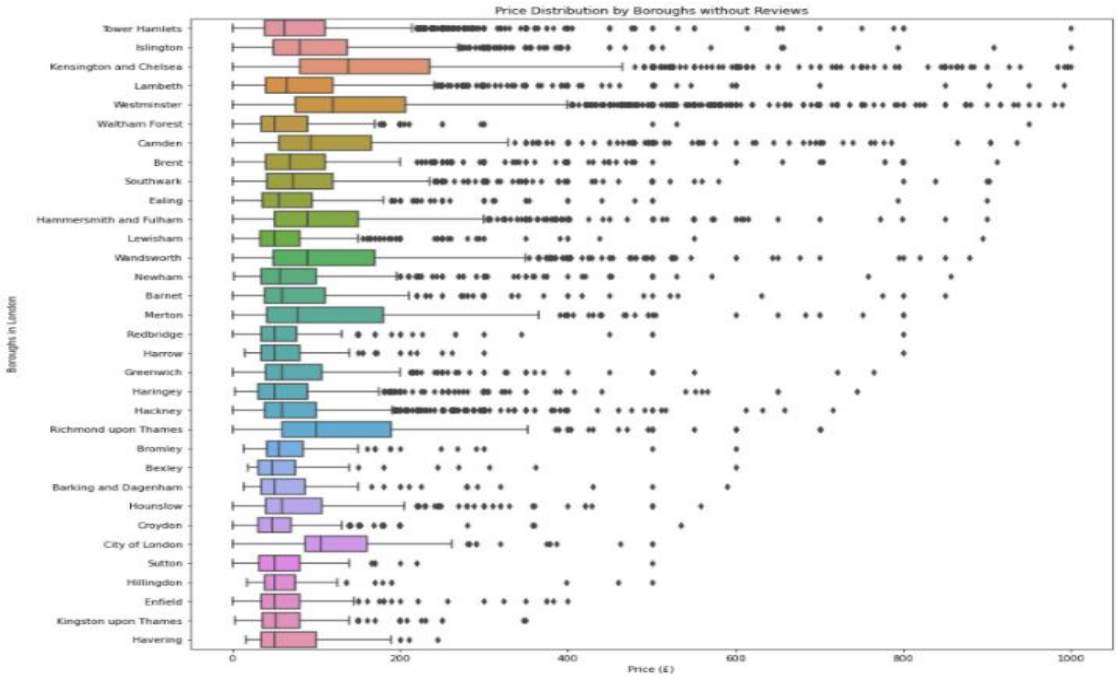


Research Question 3: For listings that did not get any bookings, what does their price distribution look like?

Fig. 5 shows the price distribution by borough for listings without reviews, that is, they had no bookings. It shows that 28.23% of the listings had no bookings. Among the boroughs, Westminster has the highest number of listings with 12.13% of listings in this borough without bookings. For room types, "private room" accounted for the highest number of no bookings, with 49% of private rooms not taken up.

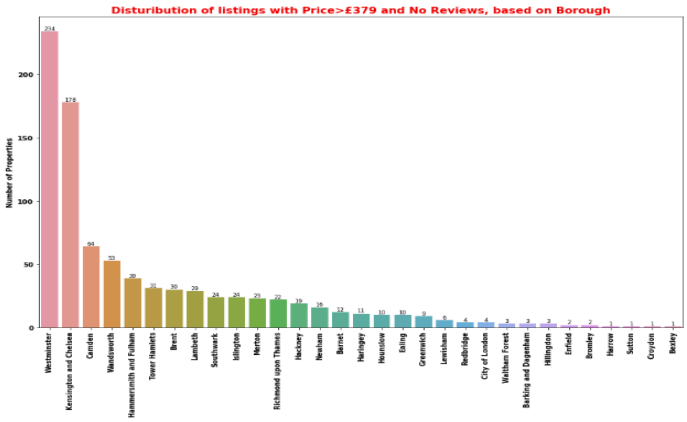
The average price for listings in this group is £13.91 more than the overall average at £113.96. Our findings show that only 4% of these listings had prices greater than four times the standard deviation. This differs from our expectations that most listings without bookings would be outliers.

Fig. 5. Price distribution for listings without bookings.



Research Question 4: Could pricing be the reason for the above listings not getting any booking?

Fig. 6. Distribution of listings with Price greater than £379



Although the average price of the listings without bookings is higher than the overall average, only 4% are in the outliers category for price. Therefore, we are not able to conclude from our findings whether pricing is the reason for the listings not getting booked.

It would appear that other factors might have an influence on the listings not getting booked.

Research Question 5: What is a good machine learning model for predicting the price of an Airbnb listing?

Based on our evaluation of the models we had selected, Neural Networks has the highest accuracy and would be a good machine learning model to predict the pricing of Airbnb listings. This is contrary to our expectation that Extreme Gradient Boosting (XGBoost) would be our best-performing model.

Conclusion:

This project has been an incredible experience for us, as we had to deal with a dataset that was rather large and noisy, and determine how to apply suitable models to produce output that was meaningful and usable. Many invaluable lessons were learned by the team including the following:

1. The importance of starting early. The team got to work almost immediately after the assignment was given, with the hope that we could finish it weeks before the deadline. As it turned out, the whole project took substantially more time than estimated, and the team worked right up to a couple of days before the deadline. Had we started later, we might not have been able to complete the work satisfactorily.
2. The modelling and results are only as good as the data we fed it. For instance, the MAE decreased by 23.7%, and the R^2 increased by 9.6% between the first and final round of data cleaning for the Linear Regression.
3. The importance of computational power. The team encountered delays and snags when trying to load files that were too big, onto the Jupyter notebook. Considerable time was expended when we ran GridSearch. E.g. we had to cancel the GridSearch for Random Forest as it was still running after 2 days. This has made us appreciate the importance of having more powerful computers.

Limitations and Further work

[1] Time constraints did not allow us to utilise interventions such as NLP to conduct sentiment analysis on the free text portions of the dataset. Had this been done, it is likely that important information could have been obtained that might affect the findings. Future research could be done in this area.

[2] Time limitations also prohibited us from merging reviews and calendar datasets with the listings datasets which would have enabled us to have a more in-depth and comprehensive analysis. This is another area for future research.

[3] This project was deliberately narrow in scope due to time limitations. Perhaps future work could incorporate data on other geographic locations in the UK, thus providing a fuller picture of pricing in Airbnb in the UK.

[4] Finally, for future work, we can consider using a more powerful computer or using the high-performance computing service provided by City called “Hyperion”. This service will allow us to run the GridSearch easily and we will be able to optimise our models with the parameters returned.

In conclusion, this project has given us a good grasp of what it takes to apply models and techniques to real-life datasets and it has been a very enriching experience for all of us.

Github Repository:

https://github.com/EamonnOCearnaigh/Airbnb-Price-Prediction

References:

Airbnb (no date) *How pricing works.* Available at: https://www.airbnb.co.uk/help/article/125/how-pricing-works?_set_bev_on_new_domain=1639858658_MTY2MzQxMTNhMzEz (Accessed: 14 December 2021).

Quinto B. (2020) *Introduction to Machine Learning. In: Next-Generation Machine Learning with Spark.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5669-5_1

Datacamp (no date) *When should I use XGBoost?* Available at: https://campus.datacamp.com/courses/extreme-gradient-boosting-with-xgboost/classification-with-xgboost?ex=11 (Accessed: 17 December 2021).

Lewis-Beck, M. S., Bryman, A., and Futing Liao, T. (2004) *The sage encyclopedia of social science research methods*, vol. 0, Sage Publications, Inc., Thousand Oaks, CA, [Accessed 23 December 2021], doi: 10.4135/9781412950589.

Livingstone, D.J., Manallack, D.T. and Tetko, I.V. (1997) *Data modelling with neural networks: Advantages and limitations*, *Journal of computer-aided molecular design,* vol. 11, no. 2, pp. 135-142.