A Comparison of Machine Learning Methods for Classification of Status Types of Facebook Live Seller Posts

Benjamin Trzcinski - 680045406

May 2021

Selling products via a live stream is quickly replacing more traditional selling methods for small businesses such as photo posts, promising more engagement and higher sales rates. This report investigates the various ways ecommerce activity can be classified as such, revealing whether the differences in metrics are enough to classify a status as a live video, or a photo. We discover that this can be predicted with a fairly high level of certainty, and the best method for this classification is through the use of a random forest classifier, providing a prediction accuracy of 86.96% when hyperparameter optimised.

1 Introduction

Live Streaming is becoming an ever more popular medium for a wide variety of activities globally[1], and as a result, there is a growing interest in the use of live streamed video to market and sell products online. This interest is particularly prominent in Asia[2][3] where live stream commerce is far ahead of the western market. The impact that video streaming has on sales for businesses is still relatively unresearched[4][5], but it is believed that the increases in real time communication and social value it provides to the consumer will likely be drivers in increased purchase intention[6][7].

The dataset used in this study is a collection of posts by a group of fashion and cosmetics retailers in Thailand, all using a split of 'traditional' Facebook strategies, as well as Facebook Live video streaming to sell their products[8]. It will be investigated whether a test dataset of these 'status types', as referred to in the dataset, can be effectively predicted and classified based of the engagement metrics provided in the assigned training data set. This will provide an insight into the contrast in viewer engagement and perceived effectiveness of live stream selling compared with photo ads, and exploring a range of different classification techniques on the dataset will reveal the differing levels of accuracy and reliability provided by each method, as well as any underlying issues and strengths of the algorithms for a dataset of this size and dimensionality.

Previous studies into this dataset have

investigated the possible long term and seasonal shifts brought about by the advent of live stream commerce in Thailand[9]. This primarily depended on the interpretation of various time series plots showing how engagement metrics changed post the release of Facebook Live. This study will build off this knowledge, working under the assumption that live video provides increased engagement metrics, as demonstrated in the previous study, and focus instead on classifying these advertisements correctly depending on the metrics provided. This will provide a further insight into the scale of the benefits of live video selling compared with other methods.

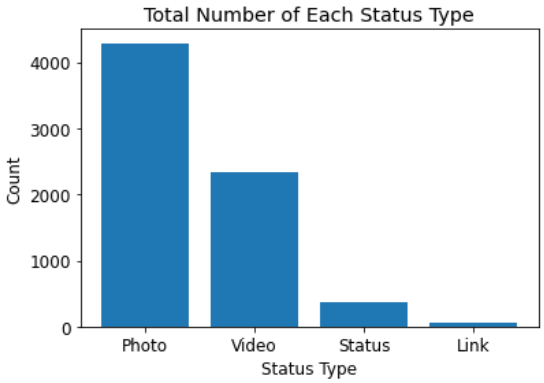

Figure 1: Total tallies for each status type in the dataset

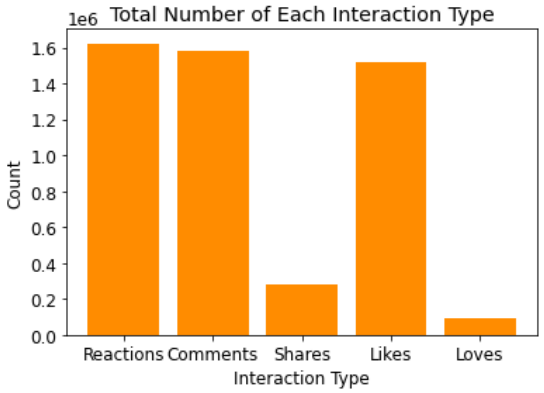

Figure 2: Total tallies for each type of engagement metric in the dataset

2 Methodology

2.1 Data

The dataset used for this analysis was collected from the Machine Learning Repository Database[8], and contains information about 7050 unique instances of Facebook use for ecommerce by 10 different businesses over the span of around 9 years. Each entry records the quantity and type of each interaction, as well as the 'status type' of each instance, referring to what the format of each entry is. The total tallies for each of these metrics are displayed in Figures 1 and 2, with 'wows', 'hahas', 'sads' and 'angrys' removed from Figure 2 due to their very low frequency (However they are still accounted for in each algorithm).

Status type will be the response variable for this study, but in order to simplify the problem to a binary classification, the status types 'link' and 'status' have been removed from the dataset, due to the relatively few instances of each. There were also 4 empty columns that required removing from the dataset before analysis, as well as the date column. This was removed because, as previously mentioned, earlier studies looked at the impact that date has on status type in detail, and so this would not be a feature of this study. Additionally, the date column contained many missing values, so removing it

allowed for the whole dataset to be used, as opposed to the ∼60% remaining when removing rows with a null date entry. Finally, the data was randomly shuffled, and partitioned with a 75/25 split for the training and test datasets respectively.

2.2 Choice of Algorithms

2.2.1 Principal Component Analysis

In order to reduce the dimensionality of the dataset and improve computation times, Principal Component Analysis (PCA) was used to project the data to 2 dimensions, while maintaining a suitable amount of variance[10][11]. This result can then be visualised to assess for any immediate trends, but more importantly, it can be applied to further classification algorithms in order to visualise groupings in a 2-Dimensional space.

The PCA function from Scikit-learn was used to carry out the Principal Component Analysis[12], using the solver type that is automatically selected to best fit the dataset. In this case, a randomized Singular Value Decomposition (SVD) method is selected, as outlined by Halko et al.[13].

2.2.2 Decision Tree Classifier

A decision tree classifier was the first algorithm tested on the dataset. It was chosen as the initial method applied as it needs fairly little data preparation, and the logical steps used can be visualised well, due to the algorithm using a white box architecture, aiding with interpretation of the classification[14]. It is also easy to apply statistical tests to the results of the algorithm, making it fairly straightforward to produce an initial value for the level of accuracy we can expect to attain, as well as the precision and recall of the classification. The DecisionTreeClassifier function from Scikit-learn was used for the classification[15].

One possible downside of decision tree classifiers is they can often overfit data due to the generation of over-complex trees that don't generalise the data well[14]. This is accounted for in the code by hyperparameter tuning the decision tree classifier, and exploring which value for the maximum depth provides the best results.

The effects of applying random oversampling were also investigated on the dataset within a decision tree classifier using the RandomOverSampler package from Imbalanced Learn[16]. This was implemented to account for the difference in size of the two classification groups in the dataset. It was not strictly needed, as both categories have a good level of datapoints, but it was useful to investigate if it had a positive impact while considering a simpler classification algorithm. When accounting for any randomness in the oversampler, or in fact any randomness throughout the investigation, the random state seed of 0 was used to ensure consistency and reproducibility.

2.2.3 Random Forest Classifier

The RandomForestClassifier function from Scikit-learn[17] was then applied to the data to explore an alternative application of decision trees. A random forest classifier is an alteration of a traditional decision tree classifier that uses a perturb-and-combine approach in an attempt to reduce the high levels of variance typically associated with decision trees[18]. They work by generating multiple random subsamples of the data,

on which decision tree classification is individually performed, and then averaging the results, cancelling out the decoupled prediction errors associated with the injected randomness, and reducing the chance of overfitting the data[18].

Random Forests can lead to an increased risk of bias affecting the results[19], but the often significant gain in variance reduction can lead to an overall better result[18]. Similar to the decision tree classifier, hyperparameter tuning was used to investigate the optimal value for the maximum depth of the random forest.

2.2.4 Logistic Regression

Logistic regression, also known as maximum-entropy classification, was the next choice of algorithm tested on the dataset. It was implemented using the LogisticRegression function from Scikit-learn[20], to perform a binary classification of the two response variables. Again, hyperparameter tuning was used to investigate the optimum performance of the model, this time when accounting for various different solver types within logistic regression. The five different solvers investigated were 'newton-cg', 'lbfgs', 'liblinear', 'sag' and 'saga'.

The 'liblinear' solver uses a coordinate descent algorithm which works by approximating the minimization for each value along a coordinate direction successively[21]. It is traditionally a good choice for smaller datasets. Since the dataset used in this investigation is neither very large or small, it is useful to investigate the performance differences in solvers designed for both large and small datasets, and is not immediately apparent without the relevant metric comparison which will perform better.

As such, the 'sag' and 'saga' solvers are designed to be efficient for larger datasets, as well as better for multinomial classification, compared to 'liblinear', as they are able to use true multinomial logistic regression as opposed to a 'one-vs-rest' approach[22]. However, these benefits are not applicable to the binary classification examined in this study, and so it is left to be seen if these solver types provide any intrinsic benefit. The 'saga' solver is designed to be a variant of the 'sag' solver that performs better for sparse datasets[22]. It will be interesting to compare the results in this study, as the dataset used contains some features which are heavily populated such as 'comments', compared with some features which contain many zero values, such as 'wows'.

The 'newton-cg' solver is designed to converge faster when assessing a high-dimensional dataset[22]. The data considered in this study is not particularly high dimension, especially when compared against the number of samples for each feature. It will therefore be useful to investigate the potential increase in result quality with other solvers compared to this one.

Finally, the 'lbfgs' solver uses the Broyden-Fletcher-Goldfarb-Shanno algorithm, an approach that also favours smaller datasets[22], which can be compared against the linear algorithm used in the 'liblinear' solver to assess whether true multinomial regression will have any impact when only considering a binary classification.

2.2.5 K-Nearest Neighbours

To perform this classification, the KNeighborsClassifier function from Scikit-learn[23] was used. It is an instance-based learning algorithm that stores instances of the training set, instead of constructing a general internal model[24]. The dataset is then classified by taking the majority vote for the assigned number of nearest neighbours for each point[24]. As such, the given value for the number of neighbours to compare can have a large influence on the performance and results of the algorithm, and so hyper parameter tuning was used to investigate the optimal value for this dataset.

Additionally, three different solver types were investigated, each using a ball tree, KD tree and brute-force search approach respectively. Although, these differences were expected to have less of an impact than the more diverse solver types present in logistic regression for example, and so an initial investigation into performance was carried out to pick a generally well performing solver, before further hyper parameter tuning was performed on just the chosen solver.

2.2.6 Support Vector Machines

The SVC and LinearSVC functions from Scikit-learn[25][26] were used to apply support vector classification to the dataset. Typically, support vector machines are effective for high-dimension datasets, avoiding over-fitting[27]. However, due to the computational expense required, Support Vector Classification (SVC) was only applied to the PCA results for the dataset. This additionally allowed for the classification regions to be plotted in 2-Dimensions, allowing for a clear comparison between the three different kernels used. But, it potentially sacrificed some of the performance that would be provided by a non-dimensionally-reduced dataset.

The three kernels considered were a 'linear', 'rbf' and 'poly', where poly had degree 3. The linear and degree 3 polynomial kernels generate fairly straightforward classification regions, only partitioning data points by straight and degree 3 polynomial lines respectively. The Radial Basis Function (RBF) kernel is a more complex approach that aims to best balance the 'C' and 'gamma' parameters for the function, providing the best possible classification regions for the data[28].

3 Results and Discussion

3.1 PCA

Figure 3 demonstrates how the data is separated after PCA is performed. Live video (or live stream) statuses will henceforth be refered to as 'videos' for simplicity.

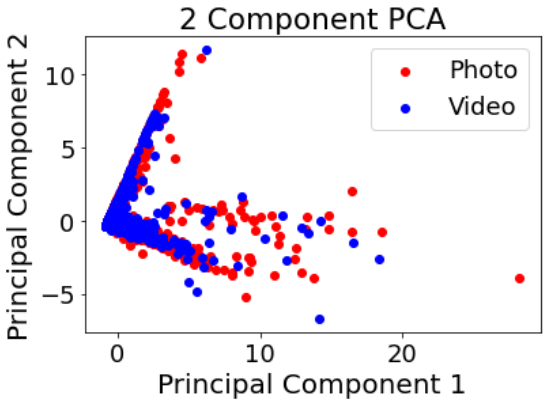

Figure 3: Labeled 2 Component PCA plot for the training dataset

There are some slight differences in the way outlier values for photo and video classifications are plotted, but in general, the PCA revealed no immediately obvious trend that could be used for classification. Almost all of the data points are in a tight band between values of -5 and 10 for each principal component, with no discernible difference between photo and video statuses.

3.2 Decision Tree and Random Forest Classifiers

When considering the results of the classifiers, it is important not only to assess the accuracy value, but also the recall and precision values. These ensure that the algorithm doesn't have a misleadingly high level of accuracy, when in actuality it is predicting many classifications incorrectly also. These values are therefore considered for every algorithm used, and used to measure the success of the algorithm while hyperparameter tuning. As a result, Figures 4, 5 and 6 were generated, showing the varying levels of the metrics for each algorithm, while varying the maximum depth value for each run.

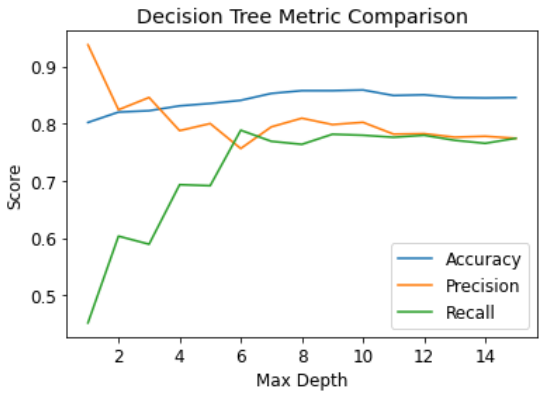

Figure 4: A comparison of the accuracy, precision and recall of the decision tree classifier with varying levels of maximum depth

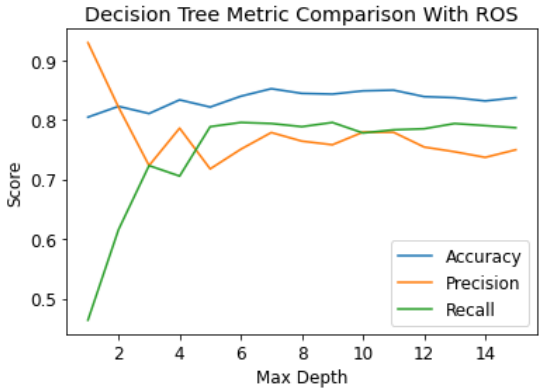

Figure 5: A comparison of the accuracy, precision and recall of the decision tree classifier with random over sampling of the data with varying levels of maximum depth

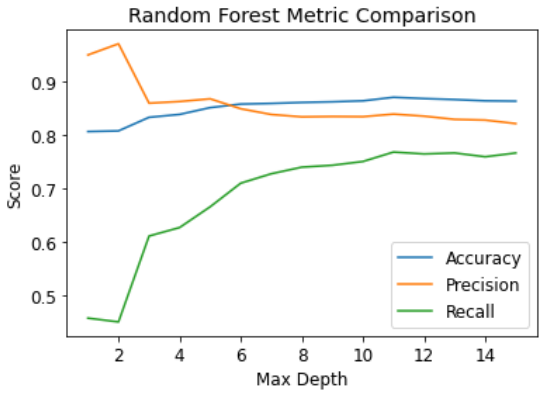

Figure 6: A comparison of the accuracy, precision and recall of the random forest classifier with varying levels of maximum depth

Using these plots, the optimal values for maximum depth for each algorithm can be identified as 10 for decision trees with and without random oversampling, and 11 for random forest. Once this has been found, a useful way to visualise the accuracy, recall and precision metrics for each algorithm with optimal hyperparameters is through the use of a confusion matrix. These plot the correctly predicted values for each response variable, as well as the misclassified predictions for each

response variable, clearly demonstrating the overall value of the algorithm. Figures 7, 8 and 9 show the confusion matrices for all three algorithms.

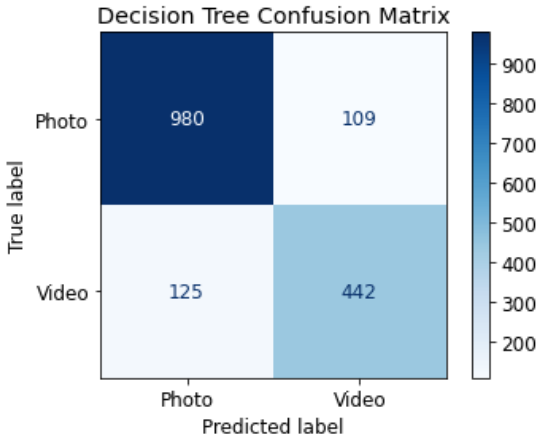

Figure 7: A confusion matrix for the decision tree classifier with a maximum depth of 10

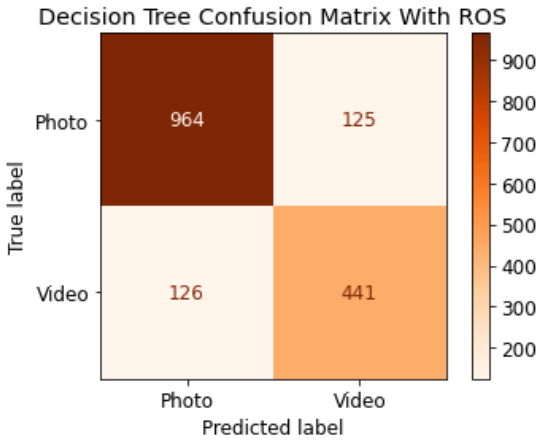

Figure 8: A confusion matrix for the decision tree classifier with random oversampling of the data with a maximum depth of 10

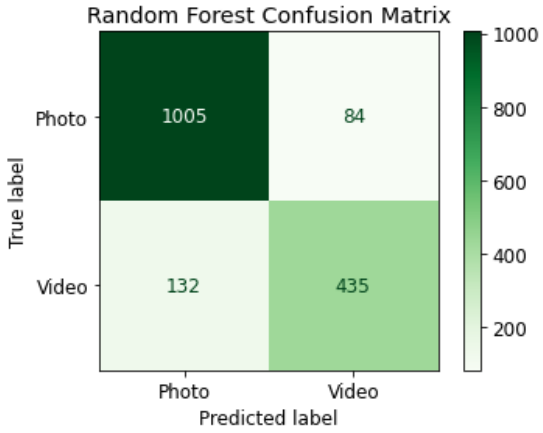

Figure 9: A confusion matrix for the random forest classifier with a maximum depth of 11

When assessing the results it is clear that the decision tree classifier performs better without random over sampling applied to the dataset. It predicts more photos and videos correctly, as well as misclassifying photos and videos less frequently. The random forest classifier is arguably even better than the decision tree classifier, predicting more status types correctly overall. However, it did falsely classify a greater number of data points as videos compared to the decision tree approach, making the comparison less straightforward than with and without the random over sampler.

In order to further compare the value of each of the algorithms, particularly comparing decision trees with no random over sampler and random forest, Receiver Operating Characteristic (ROC) curves were plotted for each of the hyperparameter optimised functions. These plot the true positive rate (recall) against the false positive rate for a range of thresholds, producing a curve with an area under curve (AOC) that approaches 1 the more accurate the prediction algorithm is[29]. Figure 10 shows a comparison of these

curves for the three algorithms tested here.

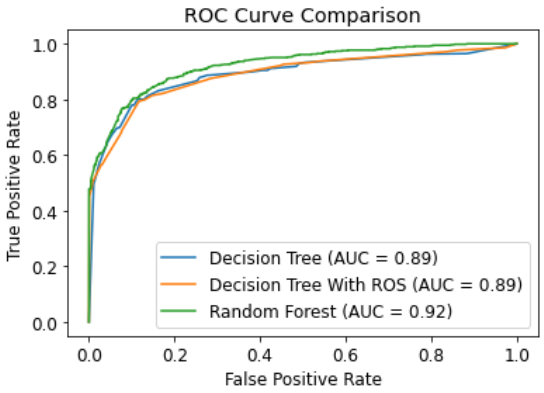

Figure 10: ROC Curves for the decision tree classifier with and without random over sampling of the data with a maximum depth of 10 and the random forest classifier with a maximum depth of 11

Comparing the curves we can see that while both the decision tree classifiers perform very similarly, achieving an identical value for AUC, the random forest classifier clearly outperforms them, achieving an AUC of 0.92, 0.03 higher than the other two algorithms. Random forest classification is therefore chosen as the optimal strategy among these algorithms, and when hyperparameter tuning, can produce a maximum accuracy of 86.96% (2dp).

3.3 Logistic Regression

hyperparameter tuning was used to investigate the best choice of solver for the logistic regression algorithm, and so similar plots were produced to the previous algorithms, comparing metrics, ROC curves and confusion matrices for each of the solvers.

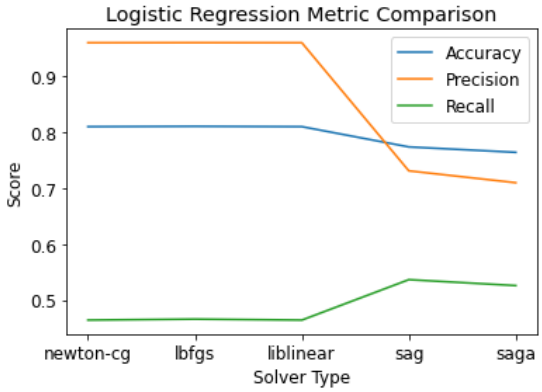

Figure 11: A comparison of the accuracy, precision and recall of the logistic regression classifier with varying solver type

When assessing Figure 11, it is clear that the 'newton-cg', 'lbfgs' and 'liblinear' solvers all show extremely similar levels of performance, with the 'lbfgs' solver very marginally improving the recall of the algorithm. The 'sag' and 'saga' solvers also show similarities, both causing a considerable reduction in precision and a slight reduction in accuracy, but a significant increase in recall. The 'sag' solver slightly outperforms the 'saga' solver in all of these metrics. The ROC curves shown in Figure 12 allow for further comparison of the models performance given the different solvers.

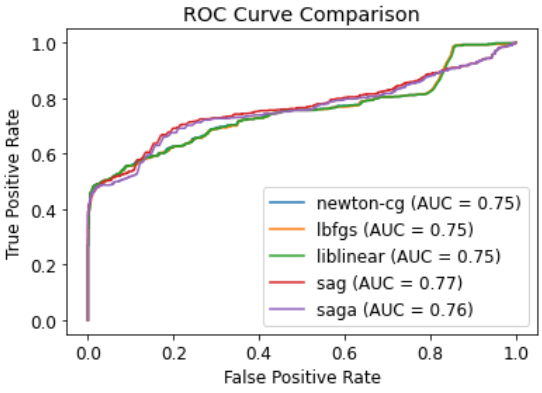

Figure 12: ROC Curves for the logistic regression classifier with varying solver type

Although resulting in lower overall levels for the majority of the metrics assessed in Figure 11, the 'sag' and 'saga' solvers outperform the other solvers here, with the 'sag' solver receiving the highest AUC value for its ROC curve of 0.77. These are considerably lower than the ROC curves calculated for the decision tree and random forest classifiers, suggesting that the logistic regression algorithm is less reliable in general for correctly classifying status types. Finally, we assess some confusion matrices for the solvers. Figures 13 and 14 show the matrices for the 'lbfgs' and 'sag' solvers, as they are identified as the best two solvers in their respective range of results.

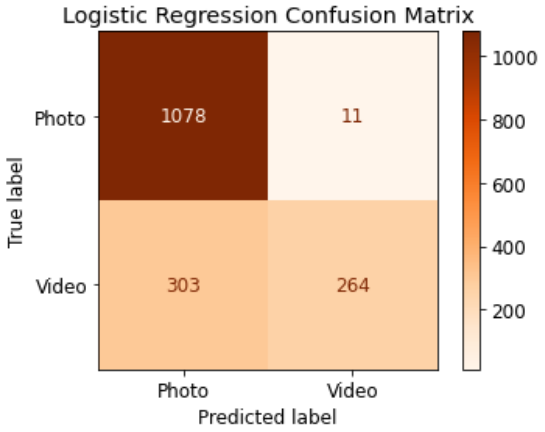

Figure 13: A confusion matrix for the logistic regression classifier with a solver type of 'lbfgs'

When assessing these matrices it appears the 'lbfgs' solver slightly outperforms the 'sag' solver, with a higher overall amount of data points correctly predicted, and a very low misclassification rate of photos. However, it struggles a lot more to predict videos, predicting more incorrectly than correctly. Overall, both solvers have value, and so the hyperparameter tuning for logistic regression reveals that both should be considered for this

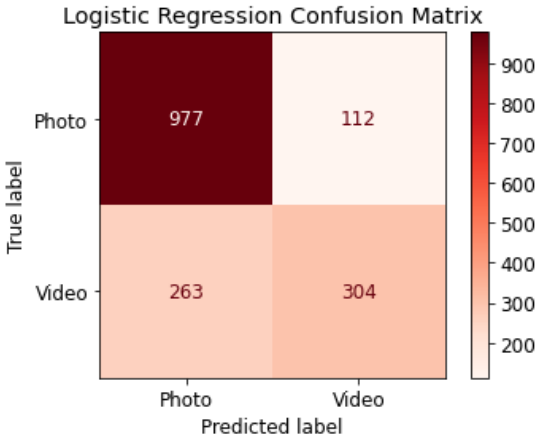

Figure 14: A confusion matrix for the logistic regression classifier with a solver type of 'sag'

application. Overall, the 'lbfgs' and 'sag' solvers predict status type with an 81.04% (2dp) and 77.36% (2dp) accuracy respectively. These are both significantly worse than random forest classification, and so logistic regression is not advisable for classification on this dataset.

3.4 K-Nearest Neighbours

Altering solver type was not initially considered for k-nearest neighbours, but once again, hyperparameter tuning was used, this time to investigate the optimal number of neighbours to consider when predicting a typing for a data point. The results of the three metrics considered for each algorithm with a varying number of neighbours is shown in Figure 15.

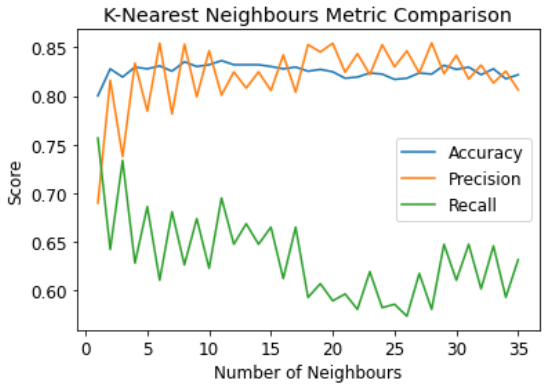

Figure 15: A comparison of the accuracy, precision and recall of the k-nearest neighbours classifier with varying numbers of nearest neighbours considered

Prediction accuracy remains fairly stable throughout, with precision and recall gradually rising and falling respectively. Although there isn't a drastic change in performance when varying the number of neighbours considered, an optimal value of 11 was identified that best maximises the three metrics. With this value chosen, the effects of using differing solver types was then investigated. The ROC curves for each of the solver types selected are plotted in Figure 16.

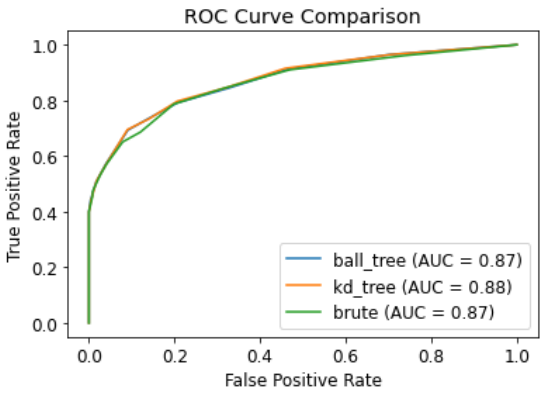

Figure 16: ROC Curves for the k-nearest neighbours classifier with varying solver type and 11 nearest neighbours considered

As expected, there is very little variation in the shape and scale of the ROC curves for each of the solver types, with the performance of the model barely affected by which solver is used. However, for a dataset of this size and dimensionality, the ROC curves reveal that a 'kd_tree' search approach is slightly more reliable. As such, this is the choice of solver used going forward. Finally, a confusion matrix is generated using the optimum hyperparameters, as seen in Figure 17, in order to compare the performance of the algorithm against the others considered in this study.

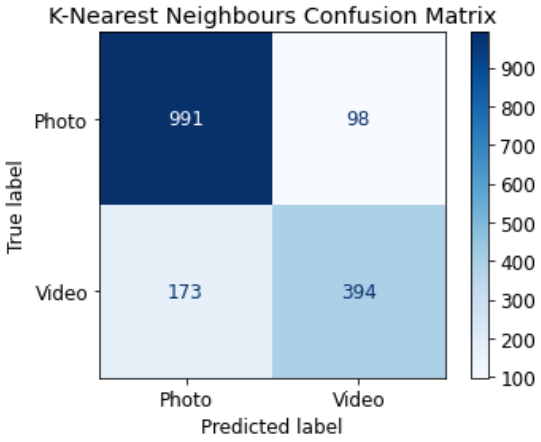

Figure 17: A confusion matrix for the k-nearest neighbours classifier with a solver type of 'kd_tree' and 11 nearest neighbours considered

Overall, with optimal hyperparameter tuning, the k-nearest neighbours algorithm has an accuracy of 83.64% (2pd). When considering this, and Figures 17, 13 and 9, it is clear that while k-nearest neighbours results in less misclassifications and a better accuracy than logistic regression, the random forest classifier is still the preferred method for classification, as it performs better than k-nearest neighbours in maximising correct predictions and minimising incorrect predictions.

3.5 Support Vector Machines

In order to make processing the support vector classification algorithm computationally viable, the PCA data was used as an input, to reduce the number of dimensions considered to 2. This further allowed for region plots to be generated, showing where different kernel types draw boundaries in 2-dimensional space in order to classify status types. These plots are shown in figures 18, 19 and 20, for the three kernel types considered, linear, degree 3 polynomial and RBF, respectively.

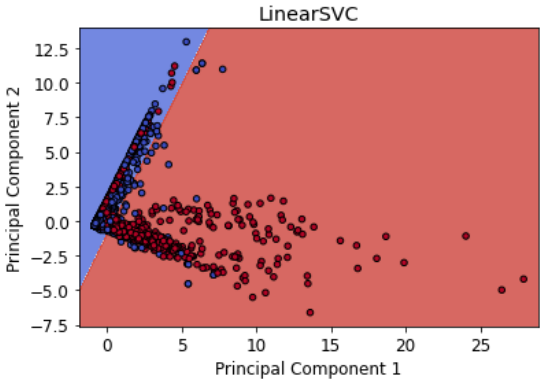

Figure 18: Classification region plot for support vector classification with a linear kernel for PCA reduced data

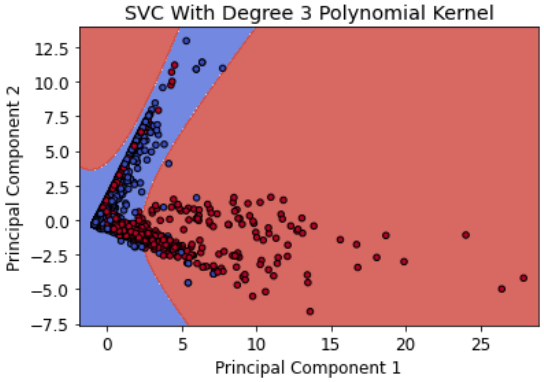

Figure 19: Classification region plot for support vector classification with a degree 3 polynomial kernel for PCA reduced data

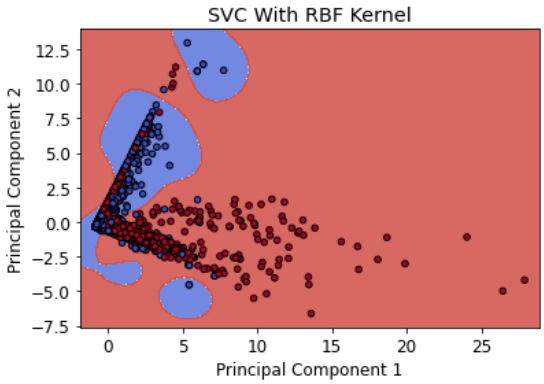

Figure 20: Classification region plot for support vector classification with an RBF kernel for PCA reduced data

At first glance it would appear that the polynomial kernel would result in more accurate predictions, due to its more malleable parameters. However, this problem is particularly suited to linear partitioning once PCA is applied, and considering the values for accuracy, precision and recall for each kernel reveals values of 78.44% (2dp), 92.34% (2dp) and 40.39% (2dp) respectively for the linear kernel, compared with values of 70.59% (2dp), 96.51% (2pd) and 14.64% (2pd) respectively for the degree 3 polynomial kernel. While precision is slightly higher, there is a significant reduction in accuracy and particularly recall when considering the polynomial kernel, making the linear kernel preferable. However, when considering the results for metrics of the RBF kernel, we see values for accuracy, precision and recall of 79.59% (2pd), 92.88% (2pd), 43.74% (2dp) respectively, performing better than the linear kernel in every metric, making it the kernel of choice for optimum results from the support vector classification algorithm with PCA.

Despite this, the accuracy of this algorithm is below that of the other algorithms

tested in this study, and the best recall value of 43.74% is significantly lower than algorithms such as random forest classification. These poorer metrics are likely due to the PCA required to run the algorithm, providing less dimensions and overall less training data to fit the model with.

4 Conclusions

This study has investigated the use of various binary classification algorithms to predict whether a Facebook seller status is a live video or a photo, depending on the engagement metrics of the status. Overall, the best method for this classification was identified as the random forest classifier, using a maximum depth of 11. This resulted in an accuracy score of 86.96% (2pd), a score much higher than randomly assigning values for photo or video, even when accounting for the difference in number of datapoints for each type (guessing 100% photo because that is the more frequent typing). Furthermore, a score of 83.82% (2dp) and 76.72% (2dp) for precision and recall respectively, demonstrate that the algorithm is making relatively few false classifications, further supporting the reliability of the predictions.

These results show that status type can frequently be successfully predicted by engagement metrics using a range of algorithms, and with some hyperparameter tuning, a random forest classifier is the best algorithm for this task.

5 Self-Assessment

I rate my work and the final result for this report 10/10.

References

[1] Pires K, Simon G. YouTube live and Twitch. Proceedings of the 6th ACM Multimedia Systems Conference. 2015;.

[2] Lu Z, Xia H, Heo S, Wigdor D. You Watch, You Give, and You Engage. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018;.

[3] Cai J, Wohn D. Live Streaming Commerce: Uses and Gratifications Approach to Understanding Consumers' Motivations. Proceedings of the 52nd Hawaii International Conference on System Sciences. 2019;.

[4] Chen C, Hu Y, Lu Y, Hong Y. Everyone Can Be a Star: Quantifying Grassroots Online Sellers' Live Streaming Effects on Product Sales. Proceedings of the 52nd Hawaii International Conference on System Sciences. 2019;.

[5] Zhang M, Qin F, Wang G, Luo C. The impact of live video streaming on online purchase intention. The Service Industries Journal. 2019;40(9-10):656-681.

[6] Wang X, Wu D. Correction to: Understanding User Engagement Mechanisms on a Live Streaming Platform. HCI in Business, Government and Organizations Information Systems and Analytics. 2019;:C2-C2.

[7] Ang T, Wei S, Anaza N. Livestreaming vs pre-recorded. European Journal of Marketing. 2018;52(9/10):2075-2104.

[8] UCI Machine Learning Repository: Facebook Live Sellers in Thailand Data Set [Internet]. Archive.ics.uci.edu. 2021

[cited 14 May 2021]. Available from: https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand

[9] Dehouche N. Dataset on usage and engagement patterns for Facebook Live sellers in Thailand. Data in Brief. 2020;30:105661.

[10] Jolliffe I, Cadima J. Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2016;374(2065):20150202.

[11] Abdi H, Williams L. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics. 2010;2(4):433-459.

[12] sklearn.decomposition.PCA — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 14 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[13] Halko N, Martinsson P, Tropp J. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review. 2011;53(2):217-288.

[14] 1.10. Decision Trees — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 14 May 2021]. Available from: https://scikit-learn.org/stable/modules/tree.html

[15] sklearn.tree.DecisionTreeClassifier — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 14 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[16] RandomOverSampler — Version 0.8.0 [Internet]. Imbalanced-learn.org. 2021 [cited 14 May 2021]. Available from: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html

[17] sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 14 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[18] 1.11. Ensemble methods — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 14 May 2021]. Available from: https://scikit-learn.org/stable/modules/ensemble.html#Forest

[19] Strobl C, Boulesteix A, Zeileis A, Hothorn T. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics. 2007;8(1).

[20] sklearn.linear_model.LogisticRegression — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[21] Wright S. Coordinate descent algorithms. Mathematical Programming. 2015;151(1):3-34.

[22] 1.1. Linear Models — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[23] sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[24] 1.6. Nearest Neighbors — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/neighbors.html#classification

[25] sklearn.svm.SVC — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[26] sklearn.svm.LinearSVC — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

[27] 1.4. Support Vector Machines — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/svm.html#svm-classification

[28] 1.4. Support Vector Machines — scikit-learn 0.24.2 documentation [Internet]. Scikit-learn.org. 2021 [cited 15 May 2021]. Available from: https://scikit-learn.org/stable/modules/svm.html#parameters-of-the-rbf-kernel

[29] Bradley A. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition. 1997;30(7):1145-1159.