CST Part II Unit of Assessment

Data Science: principles and practice

Report on the Final Practical: Analysing Clickstream Data for Online Shopping

Xiangyu Zhao

Trinity College

CRSid: xz398

3 December 2020

1 Introduction

In the final practical, I am required to analyse a clickstream dataset on an e-shop clothing website[1] between April and August 2008. I am required to interpret `price 2` as representing a user's potential willingness to pay a premium price, and build a predictive, machine learning model to predict the target value – `price 2`, so that in the future the website can be dynamically altered based on the user or their behaviour, in order to maximise profit.

2 Data Exploration

By looking at the `info()` of the dataset, I can see that there are 165,474 records in this dataset, and satisfactorily, there is no empty entry in the dataset.

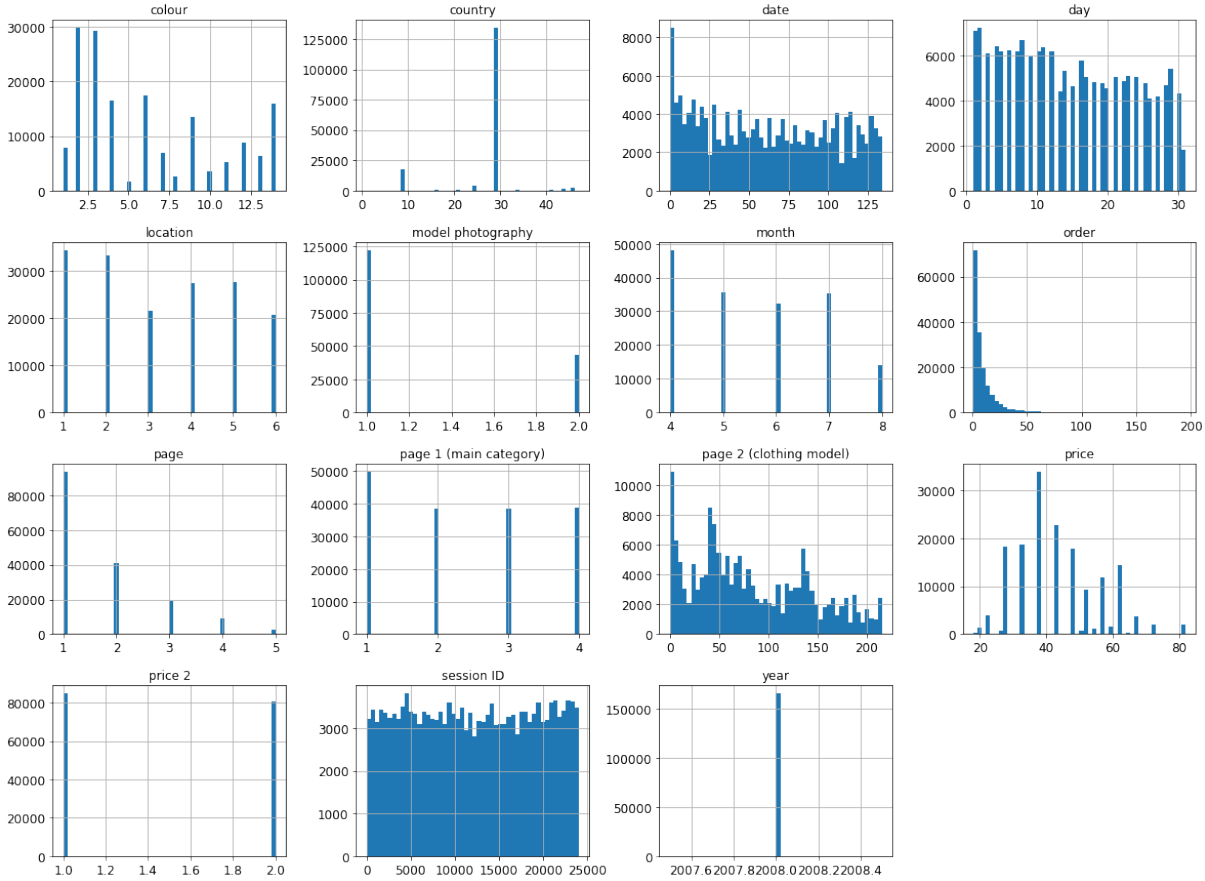
Since the `year`, `month`, and `day` attributes are separated in the dataset, I manually added a `date` attribute to the dataset that merges the `year`, `month`, and `day` attributes, setting the first day (i.e., 1 April 2008) as day 0. Since the `page 2 (clothing model)` attribute contains textual data, I encoded it into numerical values, so that I can inspect the data distribution later on.

The statistical summary of the dataset are obtained by calling the `describe()` method (excluding the `year` attribute since there is only one value):

	month	day	date	order	country	session ID	page 1	page 2
mean	5.59	14.52	+61.68d	9.82	26.95	12058.42	2.40	82.74
std. dev.	1.33	8.83	40.73	13.48	7.15	7008.42	1.14	59.12
min	4	1	0d	1	1	1	1	0
25%	4	7	+24d	2	29	5931	1	38
median	5	14	+59d	6	29	11967.5	2	70
75%	7	22	+98d	12	29	18219	3	132
max	8	31	+134d	195	47	24026	4	216

	colour	location	model photography	price	price 2	page
mean	6.23	3.26	1.26	43.80	1.49	1.71
std. dev.	4.24	1.71	0.44	12.55	0.50	0.98
min	1	1	1	18	1	1
25%	3	2	1	33	1	1
median	4	3	1	43	1	1
75%	9	5	2	52	2	2
max	14	6	2	82	2	5

The data distribution histograms of the attributes are also plotted:



By looking at the data distribution, the following findings can be obtained:

- Other than the first few days in April 2008 that has a lot more clickstream records, the number of clickstream records in the rest of the dates within the interval of this dataset are roughly at the same level, though there might be some periodic peaks and troughs;
- Nearly all sessions have no more than 50 clicks, though there exists an outlier of 195 clicks within one session;
- The vast majority of the countries recorded by this dataset is Poland (29);

- Clicks on each main category of the products are approximately the same, with clicks on trousers (1) slightly more than the others;
- There are more clicks on the top, left or middle of the web page, than on the bottom and right of the the web page;
- Clicks on the en face model photographs and on the profile model photographs have a roughly 3:1 ratio;
- Clicks on products with different prices follow a right-skewed distribution.
- More than half of the clicks recorded were on the first page of the website, and as the page number goes on, the number of clicks decays in a exponential manner;

In addition, by inspecting the definition of the target value, `price 2`, as well as other attributes, I made the following two guesses:

1. The value of `price 2` is directly derived from the mean value of price on each main category (i.e., `page 1 (main category)`);
2. The value of `price` is fixed for each clothing model (i.e., `page 2 (clothing model)`), and consequently the value of `price 2` is also fixed for each clothing model.

Both guesses are verified through coding experiments. Therefore, I can derive the following conclusion: the target value `price 2` can be directly obtained from the clothing model, `page 2 (clothing model)`.

3 Machine Learning Algorithms Implementation

3.1 Trivial Classifier

Based on the insights gained from data exploration, I can trivially build a classifier that simply calculates the main price of each main category, and compares it with the `price` attribute, achieving 100% accuracy.

However, if in the scenario where the e-commerce website owner wishes to dynamically alter the website and the prices of the products based on the user and their behaviour to maximise profit, and requires a predictive model that attempts to generalise beyond this dataset, as some features of the e-commerce website from which the dataset was derived might be changed in the future, then such a trivial classifier will be unsuitable for this task, because it was built based on the assumption that price is fixed for each clothing model. Also, to build such a predictive model, I will have to interpret `price 2` as a user's willingness to pay a premium price, rather than its original definition "whether the price of a particular product is higher than the average price for the entire product category", and the attributes that directly derive `price 2`, i.e., `price` and `page 2 (clothing model)`, must be dropped when training the model.

3.2 Preparation

For this task, I prepared 4 datasets for future model training:

1. Dataset 1: the original dataset (that is, not including the `date` attribute, which is manually added by me), dropping the `price` and `page 2 (clothing model)` attributes, as they directly derive the target value;

2. Dataset 1 (scaled): this is the same as Dataset 1, with scaling applied to each attribute, using `sklearn.preprocessing.StandardScaler`. The scaling is based on the following formula:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

3. Dataset 2: obtained from Dataset 1, dropping more less-correlated features, based on the following reasonings:

   - Since all records in this dataset are collected in 2008, I can drop the `year` feature;
   - The `month` feature also seems unimportant, but judging from the data distribution histogram for day and `date` attributes, there may be periodic rises and falls with durations of a few days, so I dropped the `month` feature, but kept the `day` feature;
   - The value of the `session ID` feature seems to be automatically generated and reveal no information of the customer, so I also dropped the `session ID` feature.

4. Dataset 2 (scaled): same as Dataset 2, with scaling applied to each attribute, also using `sklearn.preprocessing.StandardScaler`.

I have split all the above-defined 4 datasets into stratified train-dev-test sets with roughly 8:1:1 separation, obtaining 134,033 records for each the training set, 14,893 records for each development set, and 16,548 records for each test set.

Before actually training any model, I also figured out the baseline accuracy, which is a model that always predicts one class. The baseline accuracy is 51.18%, obtained by a model that always predicts `1-yes` (class `1-yes` has slightly more values than class `2-no`).

3.3 Traditional Models

3.3.1 Simple Classifiers

For simple classifiers, I tried naïve Bayes models, logistic regression models, and single-layer perceptron models. For the naïve Bayes models, I tried the `GaussianNB`, `BernoulliNB` and `MultinomialNB` variations, to compare their performances. For the single-layer perceptron models, I also tried applying the RBF kernel to the datasets, for comparisons with the single-layer perceptron models without the RBF kernel. I trained those models on all 4 datasets, except for the `MultinomialNB`, which does not accept negative values, and therefore cannot be trained using the scaled datasets. The training set accuracy and the development set accuracy of each trained model are recorded in the following table:

	Dataset	Train accuracy	Dev accuracy
Gaussian Naïve Bayes	Dataset 1	57.40%	56.75%
Multinomial Naïve Bayes	Dataset 1	50.56%	50.48%
Bernoulli Naïve Bayes	Dataset 1	51.16%	51.35%
Logistic Regression	Dataset 1	56.90%	55.94%
Single-layer Perceptron	Dataset 1	48.84%	48.65%
Perceptron with RBF Kernel	Dataset 1	51.11%	51.34%
Gaussian Naïve Bayes	Dataset 1 (scaled)	57.79%	57.11%
Bernoulli Naïve Bayes	Dataset 1 (scaled)	54.79%	54.26%
Logistic Regression	Dataset 1 (scaled)	56.64%	55.45%
Single-layer Perceptron	Dataset 1 (scaled)	48.94%	48.39%
Perceptron with RBF Kernel	Dataset 1 (scaled)	49.97%	50.42%
Gaussian Naïve Bayes	Dataset 2	57.79%	57.03%
Multinomial Naïve Bayes	Dataset 2	54.27%	54.02%
Bernoulli Naïve Bayes	Dataset 2	51.16%	51.35%
Logistic Regression	Dataset 2	56.27%	55.15%
Single-layer Perceptron	Dataset 2	53.37%	53.78%
Perceptron with RBF Kernel	Dataset 2	48.84%	48.65%
Gaussian Naïve Bayes	Dataset 2 (scaled)	57.79%	57.01%
Bernoulli Naïve Bayes	Dataset 2 (scaled)	54.47%	54.00%
Logistic Regression	Dataset 2 (scaled)	56.27%	55.19%
Single-layer Perceptron	Dataset 2 (scaled)	53.63%	53.21%
Perceptron with RBF Kernel	Dataset 2 (scaled)	54.70%	54.05%

All simple classifier models produce only barely-above-the-baseline accuracies, and some of the models even perform worse the baseline model, including:

- The `MultinomialNB` model trained using Dataset 1;
- The `BernoulliNB` models trained using unscaled datasets;
- The single-layer perceptron models trained using Dataset 1, both with and without scaling;
- All single-layer perceptron models with the RBF kernel, except the one trained using Dataset 2 (scaled).

It seems from the results that all simple classifier models perform poorly in this task, and more complex models must be used to produce helpful predictions. However, there are still minor improvements gained from a better choice of dataset – models trained using Dataset 2 (both variations) and using scaled datasets generally perform better than the same models trained using Dataset 1 (both variations) and using unscaled datasets. Therefore, I can also expect Dataset 2 (scaled) to work better on training future models.

3.3.2 Ensemble-based Classifiers

Now that the 3 types of simple classifier models tried in the previous subsection all perform poorly in this task, I tried whether a voting classifier based on the above 3 models can obtain a better performance. Based on the experience gained from training the simple classifier models, since the simple classifier models generally perform better when trained using Dataset 2 (scaled), I will only train future models with Dataset 2 (scaled) from now on. For the Naïve Bayes model, I used `GaussianNB`, since it is the best-performing Naïve Bayes model among the 3 Naïve Bayes variations. For the single-layer perceptron model, I did not include the RBF kernel, since the other classifiers do not include the RBF kernel. Both hard voting and soft voting strategies have been tried. Before carrying out training the voting classifiers, correlations between the 3 individual models are calculated to make sure that they are diverse enough, as shown in the following table:

	Naïve Bayes	Logistic Regression	Perceptron
Naïve Bayes	1.000	0.707	0.146
Logistic Regression	0.707	1.000	0.093
Perceptron	0.146	0.093	1.000

The development set accuracy of each trained model are recorded in the following table:

	Dev accuracy
Gaussian Naïve Bayes	57.01%
Logistic Regression	55.19%
Single-layer Perceptron	53.21%
Hard Voting Classifier	55.97%
Soft Voting Classifier	54.68%

Neither the hard voting classifier nor the soft voting classifier performs better than the best of the 3 individual classifiers (i.e., `GaussianNB`). It seems that voting classifiers are not helpful in improving the performance of the 3 poorly-performing simple classifiers in this task.

Now that voting classifiers using an ensemble of the 3 simple classifiers perform poorly in this task, I tried some more complex ensemble-based classifiers. I trained a bagging classifier and a pasting classifier, each with an ensemble of 500 decision trees, and each decision tree trained on 200 training instances randomly selected from the training set. I also trained a random forest classifier that trains an ensemble of 500 decision trees, and I also used both AdaBoost and gradient boosting to train a ensemble of 500 decision trees. After hyperparameter tuning, I found that the AdaBoost classifier performs best when `learning_rate=1` (when `learning_rate` is less than 1, overfitting may occur), and the gradient boosting classifier performs best when `n_estimators=488` and `max_depth=5` (when `max_depth` is greater than 5, overfitting may occur). Since gradient boosting is fairly robust to over-fitting, and a large `n_estimators` number usually results in better performance, the over-fitting effect on a large `n_estimators` is negligible; also, since there is a trade-off between `learning_rate` and `n_estimators` in gradient boosting, There is no need to apply hyperparameter tuning on `learning_rate`, now that the best value for `n_estimators` has been found.

The training set accuracy and the development set accuracy of the best of each trained model are recorded in the following table:

	Train accuracy	Dev accuracy	Out-of-bag accuracy
Bagging Classifier	79.50%	79.59%	79.39%
Pasting Classifier	79.48%	79.61%	–
Random Forest Classifier	95.81%	92.47%	–
AdaBoost Classifier	95.81%	92.14%	–
Gradient Boosting Classifier	93.61%	93.37%	–

All models perform very well, with the random forest classifier, the AdaBoost classifier, and the gradient boosting classifier reaching above 90% accuracies. It seems that the complex ensemble-based classifiers perform well in this task.

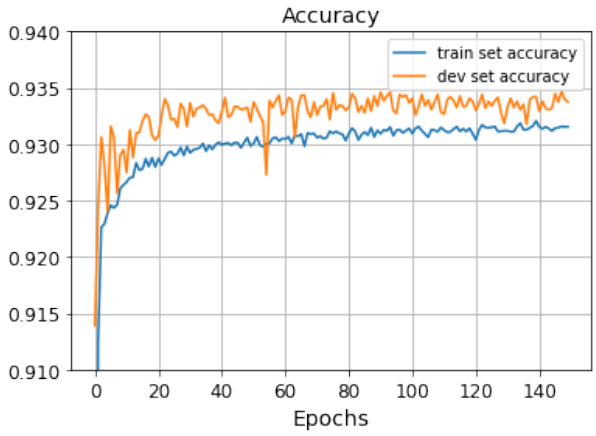
3.4 Deep Learning Models

Since this task is not a image recognition like problem with a huge number of features, or a problem that requires processing on a sequence of input, I did not try convolutional neural networks or recurrent neural networks on this task, and only trained regular deep neural networks.

The neural networks that I have built in the Practical 4 assignment served as a good starting point in my final practical. I built a neural network consisting of:

1. An input layer with 8 nodes, since there are 8 features in Dataset 2 (scaled). All nodes used ReLU as the activation function;
2. 2 densely-connected hidden layers with 64 nodes, followed by a densely-connected hidden layer with 32 nodes, and a densely-connected hidden layer with 16 nodes. All nodes used ReLU as the activation function;
3. An output layer with 2 nodes, using softmax as the activation function.

After hyperparameter tuning, the best performance of my DNN model can be reached at `epochs=150`. The training set accuracy and development set accuracy plots in the 150 epochs are shown in the below diagram:



After 150 epochs, my DNN model achieved a training set accuracy of 93.18%, and a development set accuracy of 93.38%, which is one of the best results of all models I have trained for this task.

4 Evaluation

The test set accuracies of the best of each type of models I have trained for this task are recorded in the following table (as usual, using Dataset 2 (scaled)):

	Test accuracy
Gaussian Naïve Bayes	57.86%
Logistic Regression	56.02%
Single-layer Perceptron	53.54%
Hard Voting Classifier	57.13%
Soft Voting Classifier	55.34%
Bagging Classifier	79.32%
Pasting Classifier	79.41%
Random Forest Classifier	92.14%
AdaBoost Classifier	91.90%
Gradient Boosting Classifier	91.01%
Deep Neural Network	93.37%

The DNN model is the best-performing model in terms of the test set accuracy. The confusion matrix and the precision, recall, F1 scores of the DNN model are shown below:

	`1-yes`	`2-no`
`1-yes`	7866	608
`2-no`	570	7504

Precision	92.88%
Recall	92.88%
F1 Score	92.88%

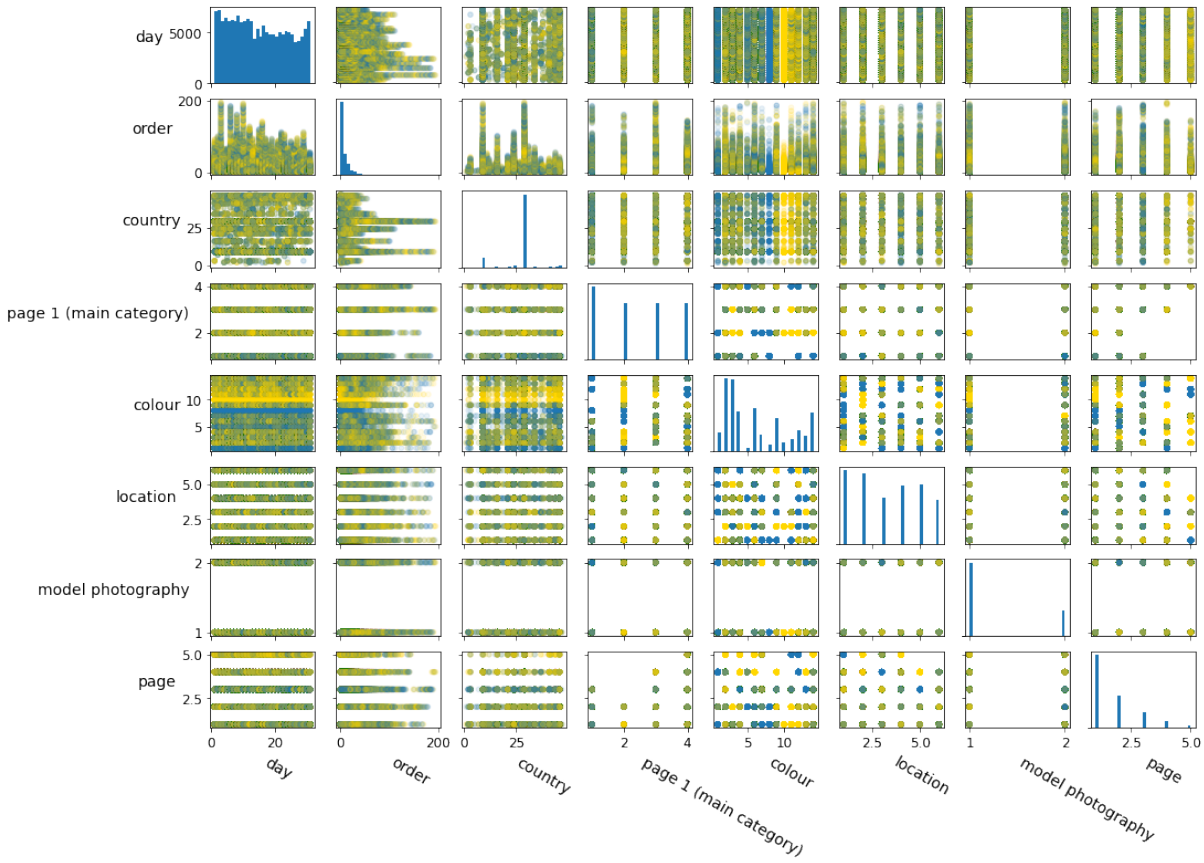
This shows that the DNN model performs evenly well on predicting both classes. Unfortunately, since there is no direct way to measure the feature importance in a neural network, the feature importance derived from the random forest classifier were used as reference, since the random forest classifier performs nearly as well as the DNN model. The sorted feature importances are shown in the following table:

	Feature importance
`colour`	33.74%
`location`	22.35%
`page`	14.58%
`page 1 (main category)`	13.68%
`order`	5.25%
`day`	4.74%
`model photography`	4.34%
`country`	1.32%

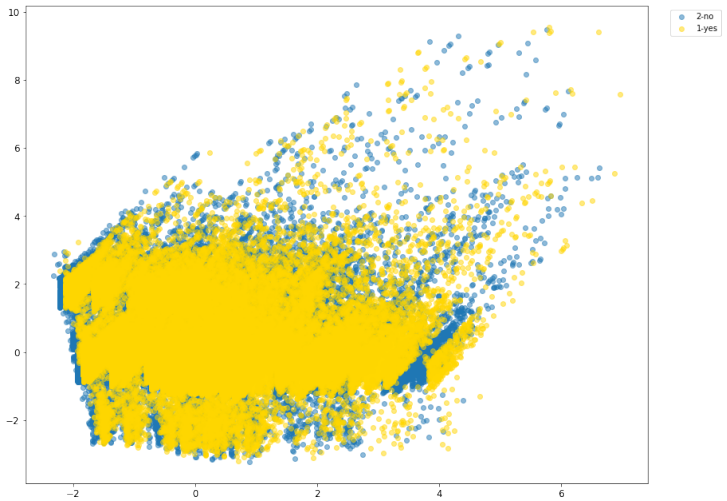
Therefore the colour of the product, `colour`, is the most important feature for predicting `price 2`. Following that, the photo location on the webpage, `location`, page number within the e-store website, `page`, and the main product category, `page 1 (main category)`, are also very important features for predicting `price 2`, compared to the other features.

5 Visualisation and Dimensionality Reduction

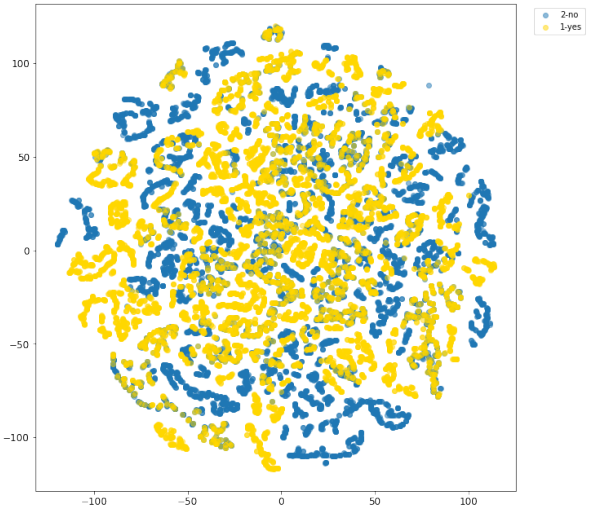
Firstly, the splom on the Dataset 2 (unscaled) is plotted, with colour-coding by `price 2`, in order to gain initial insights for further dimensionality reduction:



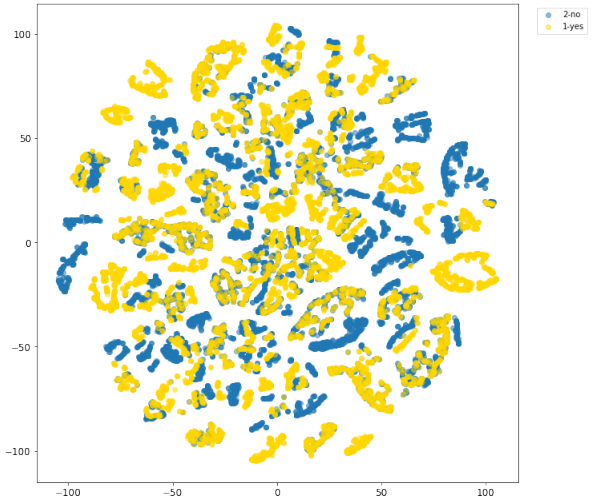
It shows on the splom that there is indeed a notable split on the `colour` attribute, which cross-validates the feature importance obtained from the previous section. PCA is then applied on Dataset 2 (scaled), and a scatter plot of the first 2 components, which represent the 2 most important features, `colour` and `location`, is shown in the below diagram:



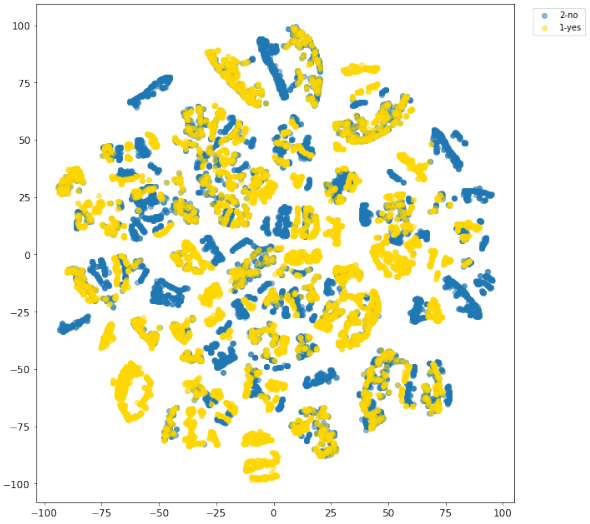
The scatters of the 2 classes almost overlap with each other, which does not provide a helpful clustering for this task. t-SNE is also applied on Dataset 2 (scaled), with different perplexities, and scatter plots of the first 2 components are also plotted, as shown in the following diagrams:



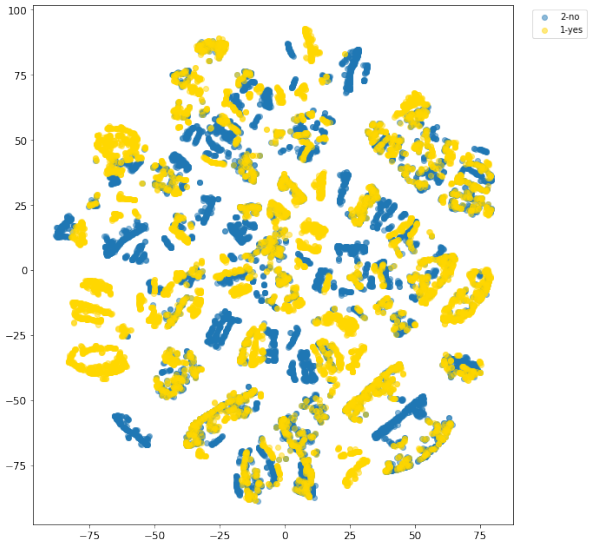
(a) `perplexity=10`



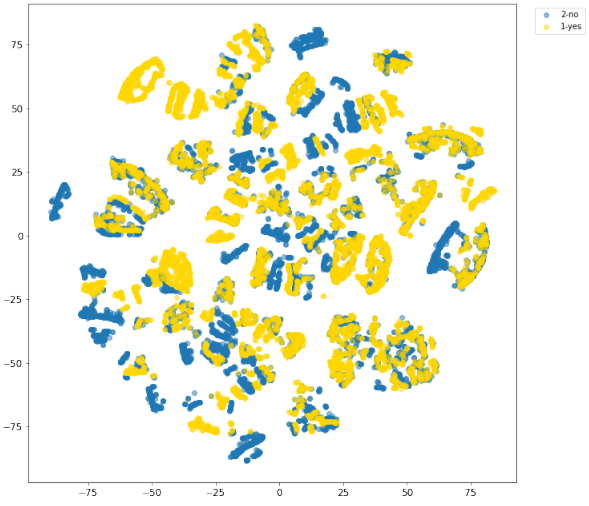
(b) `perplexity=20`



(c) `perplexity=30`



(d) `perplexity=40`



(e) `perplexity=50`

t-SNE produces a more helpful clustering visualisation. Also, as the perplexity goes up, the clustering effect becomes more and more notable, and there are less overlapping. This suggests that using t-SNE with a higher perplexity can help more in the visualisation of this task. However, there are still overlaps even when the perplexity is set to 50, which suggests that the 2 most important features alone cannot perfectly separate the two classes, and would have to rely on more features.

6 Conclusions

In the final practical, I built various models to predict the target value – `price 2`, and achieved a maximum test set accuracy of 93.37%, using a deep neural network. By inspecting on the data, calculating the feature importances from the models, and cross-validating with various dimensionality reductions, I also found the important features in predicting `price 2`, which are the colour of the product, `colour`, and the photo location on the webpage, `location`.

References

[1] Mariusz Łapczyński and Sylwester Białowąs. Discovering patterns of users' behaviours in an e-shop – comparison of consumer buying behaviours in Poland and other European countries. Sep 2013.