Experimenting with Machine Learning Models

OPEYEMI OLORUNTOLA

Word Count: 3494

EXECUTIVE SUMMARY

Data Science and machine learning play key roles in helping businesses make informed decisions based on data-driven insights. By analysing acquired past data, we extract insights, and ideas and observe patterns that enable businesses to optimize processes, customer behaviours, and marketing campaigns which helps with client acquisition and retention of businesses.

In this report, we help a (fictitious) eCommerce company fit several machine learning techniques to predict its customer income, and classify their customers based on their income, so they know how best to price their new product and decide on marketing.

We begin by exploring the data collected by the eCommerce company, analysing, and applying a set of machine learning algorithms based on features like the customers' age, how much time they spend on their website, what sector and region these customers work in, to predict estimated figures of their income using some regression algorithms. Next, we determine an income benchmark, then apply a binary classification technique to determine if a customer’s income is above or below this benchmark. We then round up by applying some clustering models to identify what features customers within an income bracket possess and try to group them based on these features.

Finally, we evaluate and recommend the algorithms that best model this dataset in predicting incomes, classifying, and clustering its customers.

INTRODUCTION TO MACHINE LEARNING.

Machine learning is a type of artificial intelligence that involves training computers to perform tasks without being explicitly programmed (Arthur, 1959)

Imagine a supermarket chain that sells thousands of goods to millions of customers daily. The number of digital records generated annually from transactions such as customer location, the total amount spent, categories of goods purchased, age and sex of customer through some checkout forms, and time spent on the website creates business questions like who are our customers? When and where to advertise? What to stock the store with? What customers buy what and at what period? What is the best price for a new product? Data starts to drive the operations of the business, not programmers (Alpaydin, 2016).

Therefore, Machine Learning is the use of algorithms to identify relationships between observations of interest which we can't explicitly program as we don't know the exact form or where relationships exist in the dataset.

Machine learning methods can be classified into

I. Supervised learning is where the output is known, and we train the algorithm to classify or predict this output accurately. The model adjusts its weights based on the input data fed into it. An example includes linear regression and logistic regression.
II. Unsupervised learning is where the output is unknown, and the algorithm finds some hidden patterns and clusters them without the need for human intervention. Examples include dimensionality reduction and clustering.

REGRESSION.

Linear regression is a type of supervised learning, where we find a linear approximation of a causal relationship between some variables and a target. It describes how a variable affects the target and is used to make predictions on estimated values. It can be used in predicting house prices given features such as the number of rooms, baths, size of the house, etc. In our case study, we use it to predict the income of customers given features like age, sex, site time, etc.

We say y is a function of X given as $y = f(x_1, x_2, x_3, \dots, x_n)$.

$$\hat{y} = X\theta$$

$\hat{y}$ to represent sample data, and y for population data.

$\hat{y}$ = target = dependent variable for which we are trying to predict accurately.

X = features = independent variables which we believe explain y

θ = quantifies the amount of a given X that describes $\hat{y}$

Methodology:

Explains the series of steps we follow to find the best model that describes our data. Following the regression pipeline, we:

i. Load the data into a DataFrame like pandas for easy data manipulation.
ii. Explore the data to ensure there are no missing values and the datatypes are consistent across columns. We observe that our dataset contains 9 columns with both numerical and categorical features which we will label encode.
iii. Declare your target and feature variables.

$$\hat{y} = Salary$$
$$X = [Age, Sex, SiteTime, \dots, Region]$$

$$\begin{bmatrix} \hat{y}^{(0)} \\ \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(m-1)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(0)} & x_2^{(0)} & \cdots & x_{n-1}^{(0)} \\ 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_{n-1}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m-1)} & x_2^{(m-1)} & \cdots & x_{n-1}^{(m-1)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{n-1} \end{bmatrix}$$

iv. Polynomial features. This is because we are dealing with a multi-regression where the power of a variable (θ_2^2) or its interaction between other variables ($\theta_2\theta_3$) may describe the target more accurately than a linear combination of that variable.

v. Standardize your data. This is to solve the problem of varying magnitudes in the features where in the age column, for example, our data goes from 19 to around 84.

$$x_s = \frac{x - \mu_x}{\sigma_x}$$

vi. Split the data into training and testing to avoid the problem of overfitting where our model learns accurately on the training dataset but fails when presented with new data.

vii. Create the regression, train your model with the training data, and make prediction on the test data.

viii. Keep optimizing and evaluating the model by experimenting with various features to reduce the cost function:
The selected cost function used is the Mean Squared Error.

$$J(\theta) = \frac{1}{m}(X\theta - y)^2(X\theta - y)$$

The R^2 is the accuracy of our model and measures the goodness of fit of a model and is mostly used as the basis for comparing models. Given as:

$$R^2 = \frac{SSR}{SST}$$

Where SSR = Variability explained by the regression

SST = total variability of the dataset.

If $R^2 = 0$, our regression explains nothing, and if $R^2 = 1$, our regression models our data perfectly.

Experimentation Notes:

1. I started by plotting all the columns against the salary to check for an obvious relationship between the features and the target.
2. Just using the numerical features with label encoded sex, and no poly fit, our linear model does 24% and a mean absolute error of 16186, and a max error of 58570
3. Next, I assumed education is ordinal, encoded it, and mapped it to our data. This reduced our mean absolute error to 15,828, max error to 56378, and an R2 score of 27% which is much better than 24%.
4. I used label encoding on the education data and this reduced the models' performance, to an R2 score of 22.9%, MAE of 16,480, and max error of 58,662.

   Based on this observation, I manually map numbers to the education data with PHD=6, and None=0.
5. I label encoded the Worktype feature, but this did not improve the model significantly, as we get a 27% R2 score and 55638 max error. However, one hot

encoding of the Worktype further reduces the max error to 54611. Less error same R2 is a better model to me.
Since we're beginning to one-hot encode our data, we'll need to implement the polynomial features which might explain variable relationships with respect to the target.

6. Our model does significantly well with an R2 score of 65%, a max error of 49966, and a mean absolute error of 9890 which is much better than all the other tests we've run.
7. I'm not convinced the region contributes to our model, but I'm going to hot encode it and add it to our feature to see how our model performs. Well, as expected, I got a negative R2 score of -2.05, a max error of 9.92e+16. However, label encoding the region improves the model R2 score to 66.7%, we get a max error of 49851 and an MAE of 9793.
8. I Increased the polynomial degree from 2 to 3 up to 10. The higher we went, the worse our model performed, leaving a degree of 2 to be the most optimal.
9. Lastly, I label-encoded all the categorical features, split my data into train, test, and validation in a 60:20:20 ratio, and ran my model again. This time I get a lesser R2 score and a reduced MSE.

I've been using SkLearn linear regression to get the tuning right. Let's test other regressions to model our data and see how well they perform.

Model Evaluation:

Model	Train Score	Validation Score	Testing Score	MSE	MAE	Maximum Error
skLearn Linear	0.732	0.629	0.66	163,153,530.831	10,322.927	43,033.349
Kernel Ridge Laplacian	0.854	0.786	0.782	104,653,082.3	7,881.056	39,440.493
Kernel Ridge RBF	0.698	0.548	0.443	267,184,812.192	12,354.677	72,572.244
Kernel Ridge Polynomial	0.817	0.61	0.549	215,988,056.506	10,982.152	78,791.434
SVM Linear	0.045	0.041	0.042	459,262,865.68	17,542.113	66,744.114

Table 1: Regressions model performance on our dataset.

Evaluation:

From table 1 above, I would recommend the Kernel ridge Laplacian regression as it models our data most accurately, not just giving us the highest score but reducing our cost function $J(\theta)$ the most. The site time feature provides no explanatory purpose to income, and we only got penalized when we used it. We got an even higher score by

just experimenting by splitting our data into training and testing sets, however, the scores reduced when we split into the train, test, and validation sets.

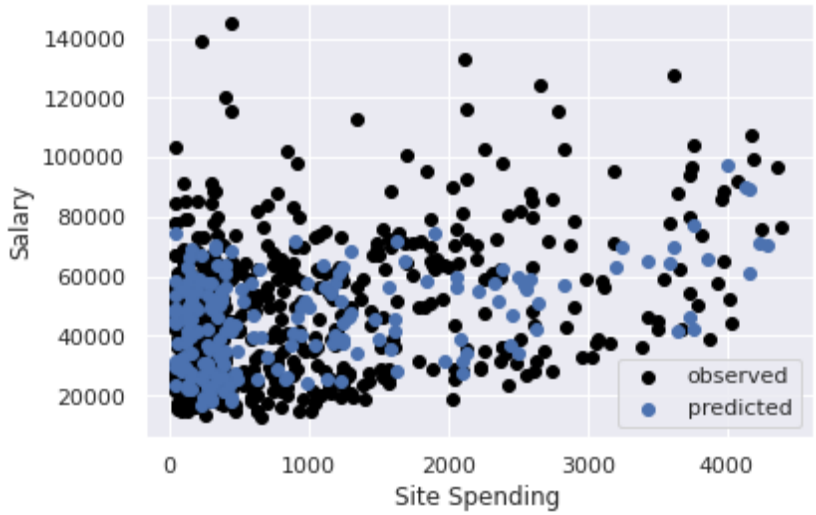

Figure 1Chart of the SiteSpending vs Salary of the observation and prediction

From figure 1 above, we can see that our algorithm does not capture the data points of the outliers even though it is the model that performed the best. Given more data, I'm sure our model will improve further.

BINARY CLASSIFICATION

In the real world, decision-making comes down to yes or no situations. If a customer will buy a product or not. Binary classification is a type of classification task in which there are only two possible classes that the data can belong. These two classes are typically represented as 1 and 0, with 1 representing the positive class and 0 representing the negative class.

In this section, we apply a binary classification technique to our dataset to determine if a customer's income is above 35,000 or under 35,000 as opposed to predicting an estimated value of a customer's income. The goal is to correctly classify data points into one of the two classes by training a classification model on a dataset that includes both positive and negative examples and then using the trained model to make predictions on new data.

Given by the equation:

$$h_\theta(x) := \begin{cases} (below\ 35k)\ if\ g_\theta(x) < 0 \\ (above\ 35k)\ if\ g_\theta(x) > 0 \end{cases}$$

The pipeline is the same as the regression except for the algorithm used in training the classification.

Methodology:

1. First, we represent the income of those above 35000 as 1 and those below 35000 as 0. I observed that the income of customers above 35000 is almost twice the size of those below 35000. This means the algorithm could label all new data as above 35000 and would be correct to a large degree.
2. To avoid this bias, I under sample my dataset by reducing the number of data points above 35000 to equal those below 35000.
3. I declared my target as the income in 0 and 1 and the rest as features.
4. To avoid overfitting, I split my data into training and testing sets without the validation as our data points are now very small and might not be enough to train the model with.
5. I standardize the features like in regression to solve the difference in magnitude problem by subtracting the mean from each data point and dividing by the standard deviation.
6. Next, I fit my classification algorithm. We would be looking at the Logistic Regression, Support Vector, and Decision tree algorithm

7. We evaluate our models and compare them for the best-performing model using its confusion matrix and accuracy.

Experiment Notes:

C is the parameter that determines the strength of the regularization where higher values try to fit the training set as best as possible and stresses the importance of each data point, while lower values try to find a coefficient vector that is as close to zero as possible. (Guido, 2016)

1. First using all the features. C=0.01, we get a 46.2% accuracy and the confusion matrix [[69, 0] [79, 0]], meaning it just predicted everything as 0 which is why we split our data to begin with, however, this did not change. So, I'll increase my c value going forward.
2. Increasing our c=0.02, we get a 62% accuracy, and a confusion matrix [[48, 21] [35, 44]]. This is better but not good enough.
3. Increasing our c = 0.1, we get an accuracy of 62% and a matrix of [45, 24] [32, 47].
4. When c = 1 to 100, we get 66% accuracy with [51, 18] [32, 47]. This is good; howe ver, the false negative is high. I'll try to use the stats model Logit to view the featur es that do not contribute a lot to our model.

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	590
Model:	Logit	**Df Residuals:**	581
Method:	MLE	**Df Model:**	8
Date:	Thu, 15 Dec 2022	**Pseudo R-squ.:**	0.1295
Time:	11:43:38	**Log-Likelihood:**	-355.94
converged:	True	**LL-Null:**	-408.87
Covariance Type:	nonrobust	**LLR p-value:**	2.692e-19

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0176	0.090	-0.194	0.846	-0.195	0.160
x1	0.1759	0.090	1.946	0.052	-0.001	0.353
x2	0.5181	0.098	5.266	0.000	0.325	0.711
x3	-0.2941	0.128	-2.302	0.021	-0.545	-0.044
x4	0.9748	0.137	7.116	0.000	0.706	1.243
x5	0.0474	0.090	0.525	0.600	-0.130	0.224
x6	-0.0492	0.091	-0.538	0.591	-0.229	0.130
x7	0.1625	0.091	1.786	0.074	-0.016	0.341
x8	-0.1503	0.091	-1.654	0.098	-0.328	0.028

Figure 2 Logistic Regression summary table from statsmodel

From figure 2 above, we can see the p-values of the features that do not contribute to our model. We remove features with values greater than 0.05.

5. Reducing our features and our c = 0.1, we get a 59% accuracy with the confusion matrix [[44, 25][35,44]], and our model does not perform better by increasing the c value.
6. Fitting other classification algorithms and checking their accuracy and confusion matrix.
7. Using the SVM algorithm with a c-value of 10, we get an accuracy of 90% and a confusion matrix of [[65, 4], [10, 69]]
8. Next, we try using the Decision tree algorithm and we get an accuracy of 90% and a confusion matrix of [[65, 4], [10, 69]] which is the same score we got from using the support vector.

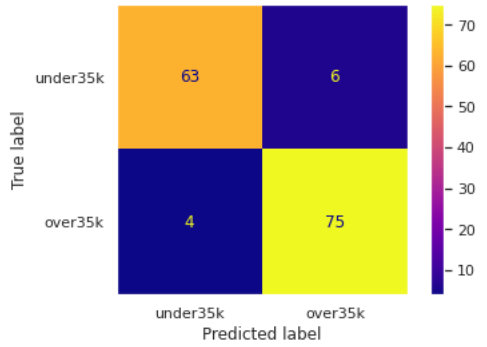

Figure 3: confusion matrix for support vector

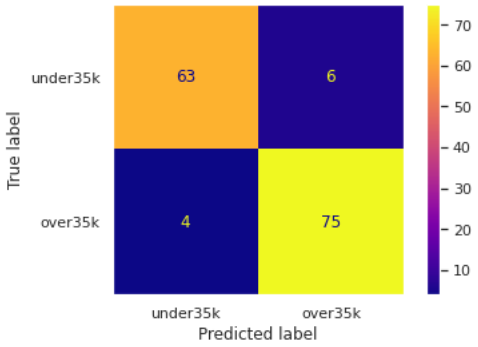

Figure 4: confusion matrix for decision tree

Model Evaluation

For this model I chose to focus more on the accuracy of confusion matrix, because our model predicting correctly meant a higher score.

Model	Training Score	Accuracy on Test
Logistic Regression	0.625	0.6014
Support Vector	0.946	0.9324
Decision Tree	0.949	0.9324

Table 2: Performance score for classification algorithms

From table 2 above, we observe that both the Support Vector and Decision Tree algorithms classify our test data with the highest accuracy. In Figures 3 and 4 above, we see the confusion matrix of both models predicting 75 correctly and missing 4 for customers earning over 35000 and 63 correctly missing 6 for customers earning under 35000. As a result, I would recommend either of these algorithms.

We calculate our accuracy with the formula:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

TP = True Positive. TN = True Negative. FP = False Positive. FN = False Negative.

NEURAL NETWORK

A neural network is a type of machine learning that tries to model the working of the brain and its neurons. In the classification task performed above, the classification algorithms used are good at modeling linear relationships and binary classification tasks however, they are not very good at modeling complex, nonlinear relationships. Neural networks can learn and improve automatically and can model high-dimensional data better than the other classification algorithms we have used. It takes input data, processes it through its nodes called neurons using the weights and biases of the features, and gives us an output.

Given by the equation:

$$a^{out^T} = f(b^T + a^{in^T} w)$$

Where a^{in^T} = the input row vector of the layer

a^{out^T} = the output row vector of the layer

w = the kernel parameter of the layer

b^T = the bias parameter of the layer

f = the activation function of the model.

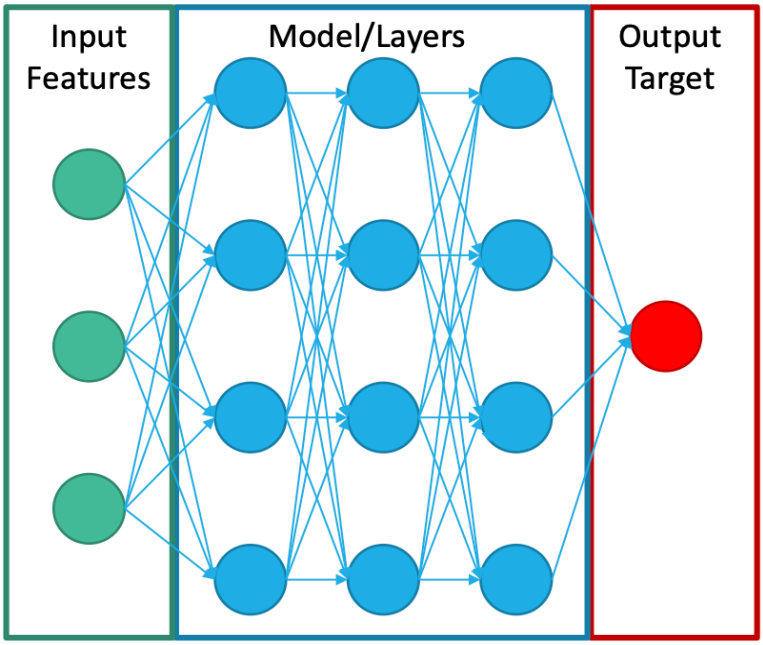

Figure 5 Neural Network Diagram: source: moodle, lecture on neural network by Peter Soar

Methodology:

1. Takes input data.
2. Fits the algorithm to model the data. In this case, classification.
3. Applies an objective function, which is a measure used to evaluate how well the models' output matches our desired output. Like in other algorithms where we try to minimize our loss function.
4. Applies an optimization algorithm by varying the models' parameters until we get the smallest loss function.

Experiment Notes:

1. I declare my number of inputs as 8, as I'm using all the features, and output as 2 since it's a binary classification.
2. I decided to split my data into training, testing and validation set even though the dataset is small with the assumption that the neural network will outperform the other classification algorithm.
3. Define our layer, using the Adam optimizer, L2 norm as the objective function, and cross-entropy as a loss.
   $L2 = \sum_i (y_i - t_i)^2$ which is our objective function i.e., the sum of squares of the output y from the target t.
   $L_{(y,t)} = \sum_i t_i \ln y_i$ which is the cross entropy i.e., the sum of the target multiplied by the logarithm of the output
4. Then I set the epoch and batch size to 100 and run my neural network.
5. The lower the loss, the higher the accuracy until at some point when the loss started increasing again. I then added the early stopping function with patience of 2 degrees so that when the loss starts to increase the model breaks out after 2 observations.
6. Run the algorithm again, and I get a training accuracy of 93%, however the confusion matrix was very inaccurate as seen in figure 6 and 7 below.
7. Using the exact dataset as the binary classification used in the previous section and gradient descent optimization, I get an 80% accuracy and a more accurate confusion matrix as seen in figure 8 and 9 below.

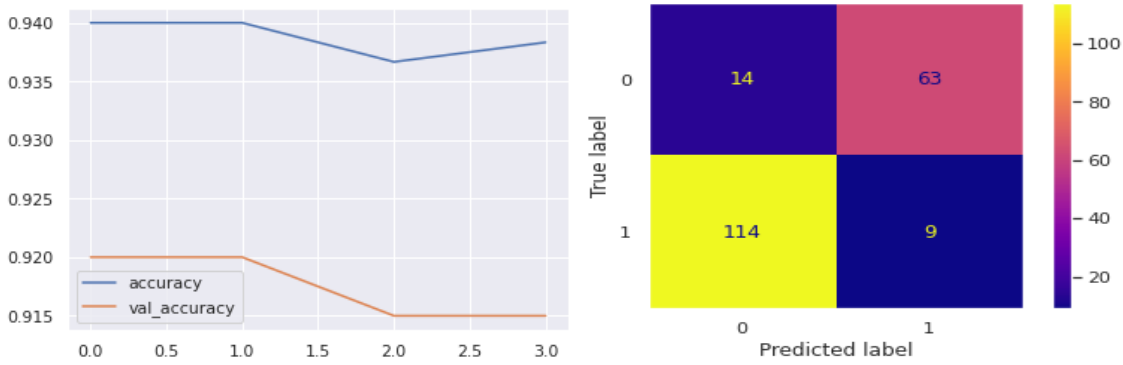

Figure 6 Plot of the high accuracy wrong classification

Figure 7 Terrible confusion matrix with high accuracy.

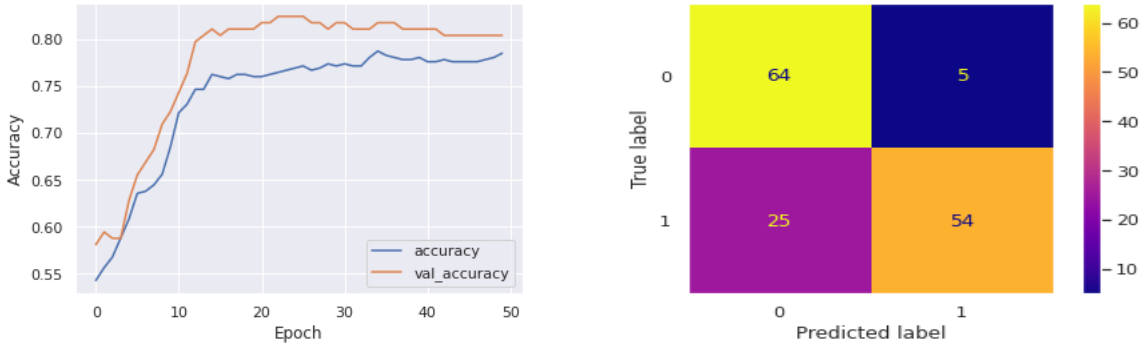

Figure 8: Accuracy plot of the newly trained model *Figure 9: Confusion matrix of the new model*

Evaluation:

Given more time, I would have liked to tune the different parameters for the layers and try other optimizers and objective functions. I used more inputs than I did with the classification too. However, with the current parameters, our model has a 80% performance on the test data with a confusion matrix that is not as good as that of the decision tree.

CLUSTERING

Clustering is an unsupervised learning where we group data points based on their similarities and dissimilarities. Unlike in classification where we know the data points and are trying to train the machine to be able to predict accurately, in clustering we do not know what the labels are and try to identify the underlying patterns in the data.

In this data sample, we apply the k-means clustering algorithm to our data by dividing our data into clusters and where each cluster is represented by a centroid. The centroid is the mean of all the points in the cluster given by the equation

$$\sqrt{\left\{\sum_{\{i=1\}}^{n}(x_i - y_i)^2\right\}}$$

$$x_i = \text{datapoint}, y_i = \text{centroid and } n = \text{number of features.}$$

Methodology:

1. We assign the random value of k.
2. Then we select an initial cluster of centroids
3. Calculate the distance between each object to each cluster centroid.
4. Assign each object to the centroid closest to it.
5. Compute the new centroid for each cluster until there is no change in the clusters.

Experiment Note:

1. I started with k = 2 clusters plotting just the site spending against the salary as that seems like an obvious point of interest, but I immediately observe that some unexplained clusters looked like outliers that the model was not capturing.
2. I applied the elbow method in selecting the optimized number of clusters. By calculating the within center sum of squares and as until there is no change in the cluster. This gave me a cluster k = 4 as seen in figure 10 below.

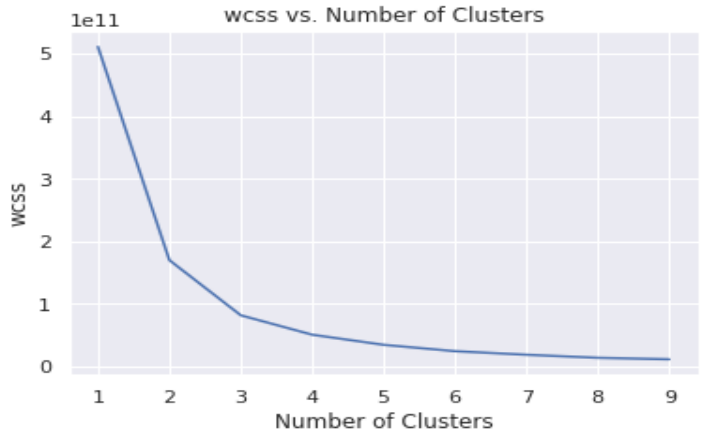

Figure 10: Elbow decay to determine cluster number.

3. I applied the k-means clustering model and plotted my data which did not explain a lot. From figure 11 below, we cannot explain the outliers as I expected it to cluster the big spenders together and the little spenders together.

Figure 11 Cluster of salary and site spending

4. I standardized my data and then applied the model again, this time the graph seemed to capture the data I had concerns about however, it does not seem like an optimal model for me as the customers that spend so little are not completely isolated as in figure 12.

Figure 12 Cluster of salary and site spending (standardized)

Evaluation:

We tried to identify clusters and underlying patterns in our dataset. However, the K-means clustering algorithm does not segment our clusters accurately. We will need to try others.

CONCLUSION

This report discusses the use of machine learning techniques to classify and predict the income of customers for a fictional eCommerce company. We begin by introducing machine learning and differentiating between supervised and unsupervised learning techniques. The focus of the report is on regression, a type of supervised learning, and how it can be used to predict income based on features such as age, website usage, and sector of employment. The report then discusses the methodology used to fit several machine learning algorithms to the data and evaluate their performance, including linear regression, logistic regression, neural network, and clustering. We conclude by recommending the best algorithms for modelling the data and providing suggestions for future work.

Works Cited

Alpaydin, E., 2016. *Machine Learning, The New AI.* Cambridge, MA: MIT Press.

Brown, S., 2021. *Machine learning, explained.* [Online]
Available at: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained
[Accessed 13 12 2022].

IBM Cloud Education, 2020. *Machine Learning.* [Online]
Available at: https://www.ibm.com/uk-en/cloud/learn/machine-learning
[Accessed 13 12 2022].

SciKit Learn, 2022. *Comparison of Kernel ridge regression and SVR.* [Online]
Available at: https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_kernel_ridge_regression.html#sphx-glr-auto-examples-miscellaneous-plot-kernel-ridge-regression-py
[Accessed 13 12 2022].

Pramoditha, R., 2021. *Encoding Categorical Variables: One-hot vs Dummy Encoding.* [Online]
Available at: https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db
[Accessed 08 12 2022].

ALLISON, P., 2012. *When Can You Safely Ignore Multicollinearity?.* [Online]
Available at: https://statisticalhorizons.com/multicollinearity/
[Accessed 08 12 2022].

Guido, A. C. M. a. S., 2016. Introduction to Machine Learning with Python. In: D. Schanafelt, ed. *Introduction to Machine Learning with Python.* Sebastopol, CA: O'Reilly Media, Inc.,, pp. 57-58.