Contents

[1. Supervised or unsupervised learning 4](#_Toc186403632)

[2.Feature selection and hyperparameter tuning. 4](#_Toc186403633)

[Hyperparameter tuning 6](#_Toc186403634)

[3. Training and testing 6](#_Toc186403635)

[4. Training and testing outcomes 7](#_Toc186403636)

[Bibliography 8](#_Toc186403637)

[Github link 8](#_Toc186403638)

# Supervised or unsupervised learning

Both unsupervised and supervised learning are good ways to tackle the given problem.

The main difference between the two is the use of labels. Supervised learning needs to know what each observation represents, what the label for a piece of data is. For example, a 50-year-old rented a standard bike in London could be the observation, and the label could be that they were riding it for 35 minutes.

With enough data and using supervised learning, we would be able to predict that for the next 37-year-old, who rents an electric bike in New York, the time they will ride the bike is 57 minutes.

Given the business problem, supervised learning would be useful to predict customer satisfaction, the cost of the bike (and therefore revenue) or the time and distance the bikes will do – which could be useful for fleet management.

Its main shortcoming is that it cannot predict anything outside the target variables for which it was trained.

Unsupervised learning does not need labelling. With unsupervised learning, the model finds hidden patterns in the data without any specific outcome (or label). In this business problem, unsupervised learning could be useful to see different customers’ segments and how to appeal to them best.

The main issue with unsupervised learning is the lack of a clear target. The patterns it discovers may not be useful for decision making. Also, evaluating the quality is not possible since there is no labeled data to compare these patterns with.

The best option would be to use both supervised and unsupervised learning. However, due to time constraints, only supervised learning will be used. The reason is that supervised learning provides predictive capabilities that can help with forecasting and can be easily interpreted and evaluated.

# 2.Feature selection and hyperparameter tuning.

With Rider Satisfaction as my target variable, I performed data preprocessing and feature engineering. I proceeded to do feature selection. To do this I used a random forest classifier and class weights:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

However, the precision is very low (similar to random chance) which doesn’t make me select any feature. I thought this could be due to skewness in the data, so I reduced it and used different scalers. However, this did not improve the results.

I fine tuned the parameters to raise the accuracy. Since the model is underfitting, I needed to make it more complex. Therefore, I used Random Search; a full search would use too many computing resources.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamenteThe results were unfortunately still the same, so I changed the method.

I opted for RFE (Recursive Feature Elimination). This method trains the model by using all the features and then eliminates the less important ones. It does this recursively, getting a ranking of the features that are the most important. I used this with Logistic Regression as the model. However, the precision of the model was even less than the Random Forest. Therefore I also discarded this approach.

Due to very low precision when predicting Rider Satisfaction, I changed the target variable to Ride Duration. I binned the variable in 20 minutes chunks. I used the Random Forest and set a limit at 0.01 importance. In this case, the precision was acceptable. With this, I concluded the feature selection process. Please see the features and their importance on the previous image.

## Texto Descripción generada automáticamenteHyperparameter tuning

Apart from the previously mentioned RandomSearch, I used a grid search to get the best model and hyperparameters.

Logistic regression, decision tree, random forest, k-nearest neighbors, support vector machine and neural network were evaluated.

Texto

Descripción generada automáticamente

# 3. Training and testing

As instructed, I trained the models for two different splits (0.15 and 0.25) . I scaled the numerical features so that the performance of the models relying on this weren’t affected. Also, I used resampling through SMOTENC for the minority classes since I have both nominal and continuous features.

During the training, I used cross-validation with 10 folds, which divides the training data in 10 parts. Then it uses 9 for training and 1 for testing.

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

# 4. Training and testing outcomes

After the training, the results for the first split are the following:

Tabla

Descripción generada automáticamente con confianza media

For the second split, they are the following :Tabla

Descripción generada automáticamente

I can see that most of the models are overfitted, except Logistic regression and Neural network. Between them, neural network has the best accuracy, in particular in the first split. Neural network is using logistic activation functions as well which could explain why both of these succeed.

It can also be seen that the other models have even greater accuracies, but only on the training data. This means they are overfitting.Tabla

Descripción generada automáticamente

Decision Trees and Random forests have great accuracies for the training data, but they fail to generalize when evaluated on the test data. They share a tree structure, which could be the cause of this.

# Bibliography

Harris, C.R. *et al.* (2020) ‘Array programming with NumPy’, *Nature*, 585(7825), pp. 357–362. Available at: https://doi.org/10.1038/s41586-020-2649-2.

Pedregosa, F. *et al.* (2011) ‘Scikit-learn: Machine Learning in Python’, *Journal of Machine Learning Research*, 12(85), pp. 2825–2830. Available at: http://jmlr.org/papers/v12/pedregosa11a.html.

The pandas development team (2020) ‘pandas-dev/pandas: Pandas’. Zenodo.

Waskom, M.L. (2021) ‘seaborn: statistical data visualization’, *Journal of Open Source Software*, 6(60), p. 3021. Available at: https://doi.org/10.21105/JOSS.03021.

# Github link

Please note that I mistakenly worked on another GitHub link and therefore the class link does not reflect all the commits I have done.

Class github: <https://github.com/CCT-Dublin/ca2-SanchezJoseAntonio?tab=readme-ov-file>

Previous github: <https://github.com/SanchezJoseAntonio/BikeRentalAI>