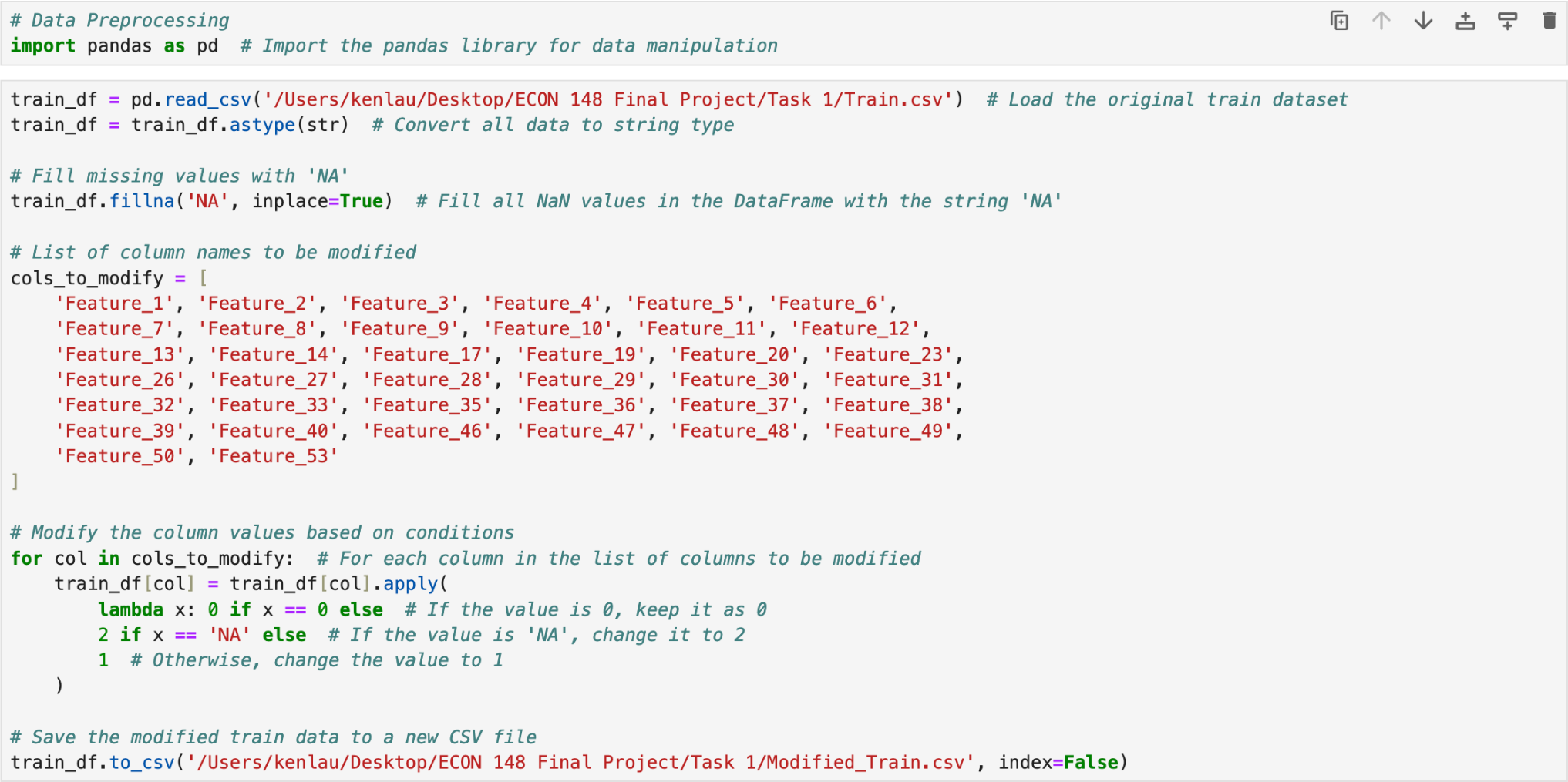
ECON 148 Final Project Task 1 Report - Ken Lau, Jake Beaney

When first looking at the training dataset, we realized that a lot of data was missing and there was no specific description of each feature. The initial thought was to process the data into numerical and categorical categories and then use the Random Forest to train the model.

However, even though we assembled GridSearchCV into the model training process, the final F1 score we got with the best parameters was less than 0.1, which indicates that the performance of the model is awful. After a long time of discussion and research, we made sure that the most important part of this project was data preprocessing.

1. Data Preprocessing

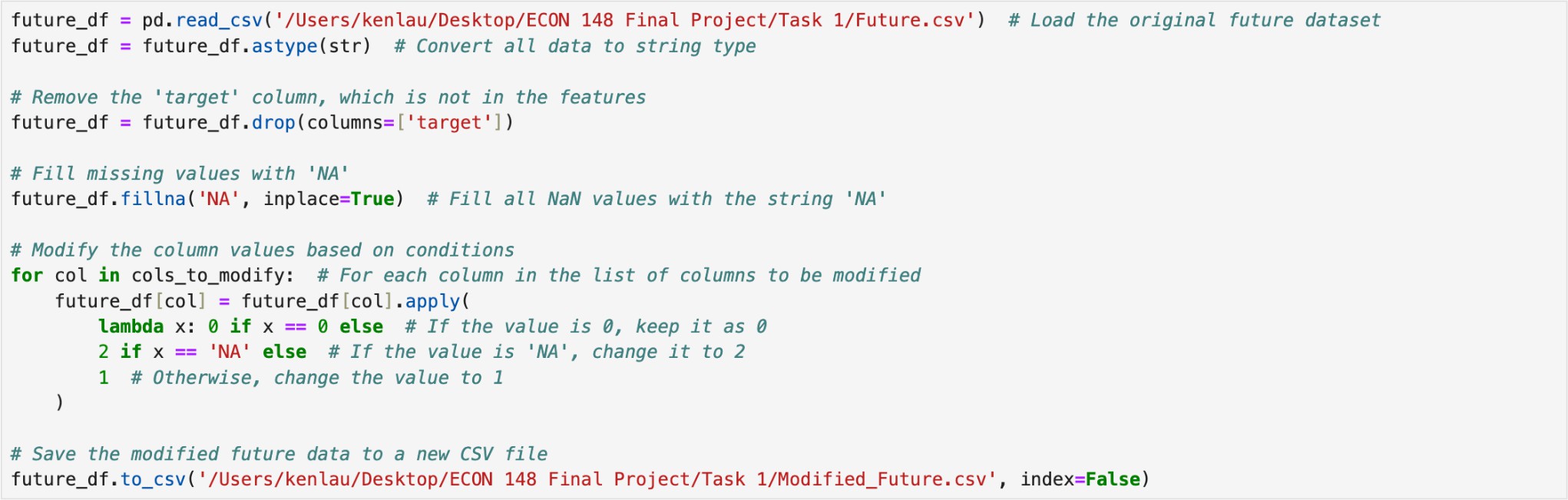


We first load the original training dataset from the CSV file and then fill all the NaN values in the dataset with the string ‘NA’. For most of the columns in the dataset, the

non-missing values are numerical values. It takes really long to train the model and also the performance of the model would be bad because mixed-type columns would lead to errors and

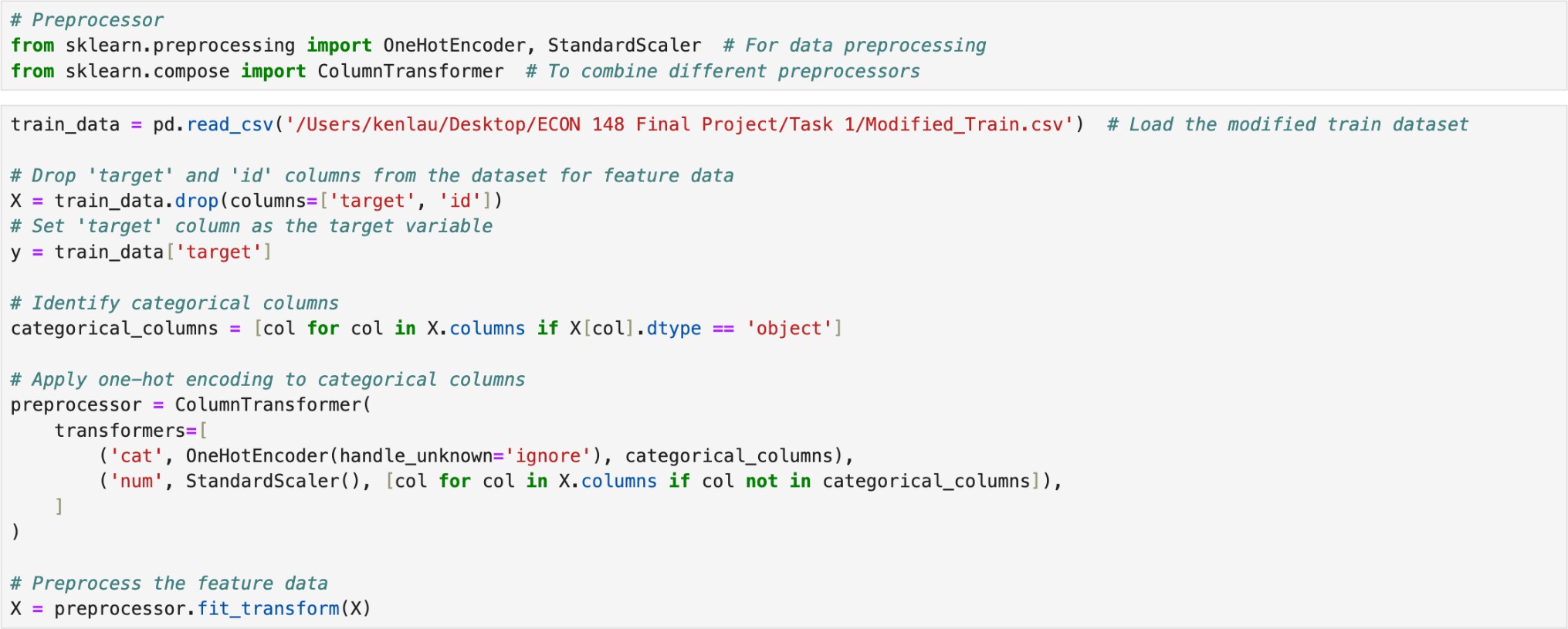
significantly decrease the efficiency of the model. So, we want to make all the numerical columns into categorical columns.

We modified the values in each column, changing the ‘NA’ cells to ‘2’, ‘0’ cells to ‘0’, and all the other cells to ‘1’. The approach not only reduces the complexity of the data but also leads to faster training and testing time.



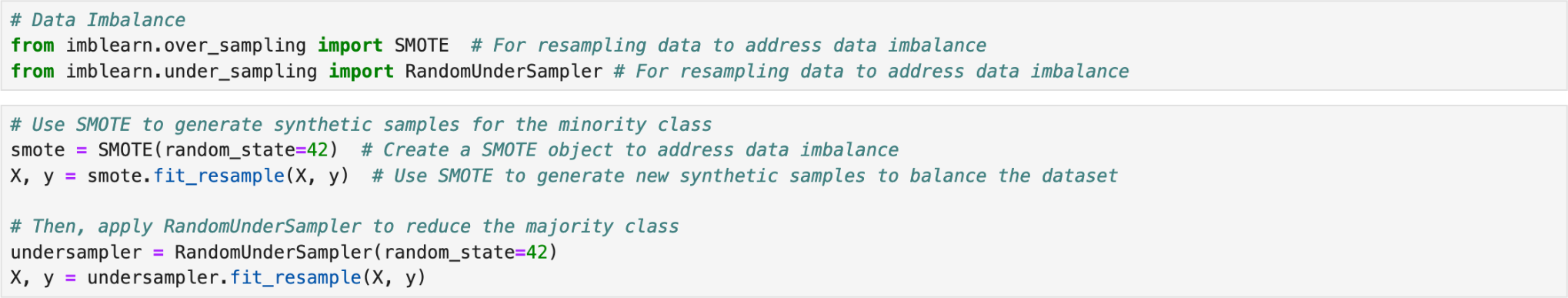
After the modification, we saved the training dataset to a new CSV file to check the characteristics of the training data, and we did the same for the future dataset and saved it.

1. Preparing Preprocessor for Data



We load data from the modified training dataset and set all the columns beside the ‘target’ column and ‘id’ column into X and the ‘target’ column into y. We then identify the categorical columns and apply OneHotEncoder to them. All the other columns are identified as numerical columns and we use StandardScaler to it. OneHotEncoder prevents algorithms from misinterpreting categorical data as ordinal or numerical. This reduces the risk of algorithms deriving false relationships between category values. StandardScaler standardizes numerical columns, adjusting them to have a mean of 0 and a standard deviation of 1. So both approaches together bring a more clean and encoded dataset for model training.

1. Dealing with Data Imbalance



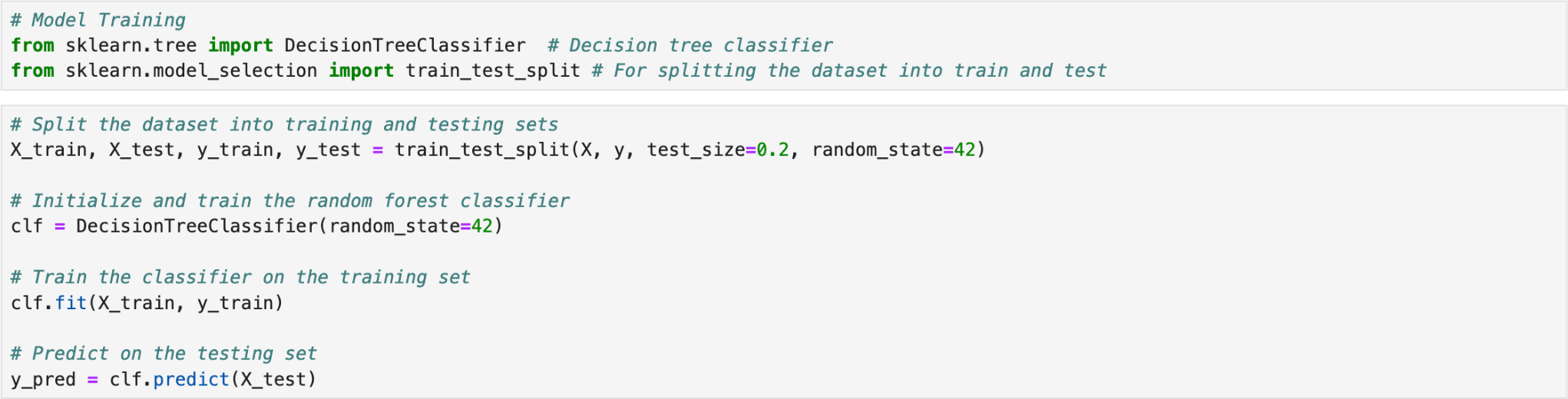
We use SMOTE to generate synthetic minority class samples to address category imbalance since ‘0’ is in the majority, and ‘1’ is in the minority so the data is imbalanced. Data imbalance is a common challenge when dealing with classification problems, where some categories have far fewer samples than others. This may result in the model not learning enough about the minority categories, thus affecting the performance of the model. SMOTE is a commonly used method for solving the category imbalance problem by generating synthetic minority class samples.

We then use RandomUnderSampler to reduce the majority class. Random undersampling involves removing a portion of the majority class to achieve the desired balance with the

minority class. It helps to counteract the effect of SMOTE creating an overly large dataset with many synthetic samples while still retaining a representative distribution of the classes.

Now we have finished all the data preprocessing jobs and we are good to train the model.

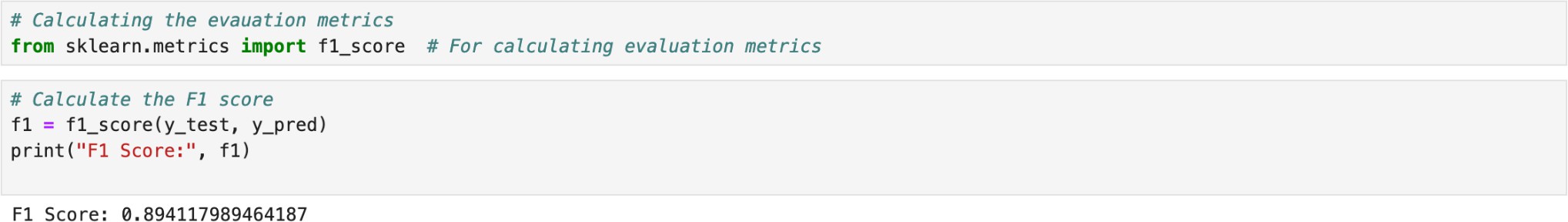
1. Model Training



We selected the Decision Tree Classifier as the model since it is good at datasets with a lot of features and it reduces a lot of time compared to Random Forest Classifier. We split 80% of the data for model training and 20% of the data for model evaluation.

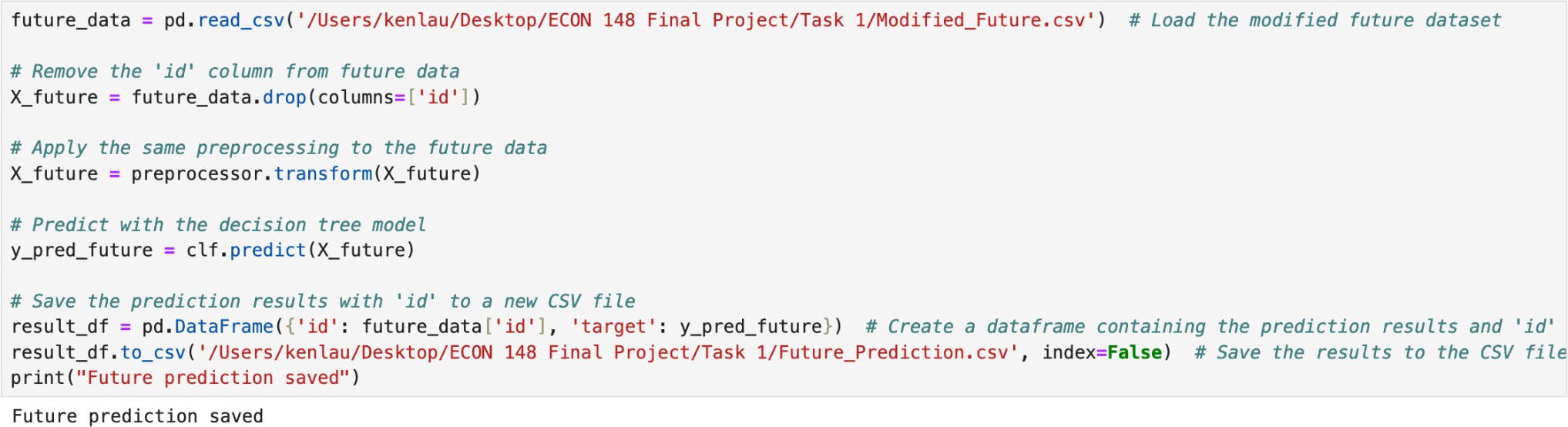
We tried to use GridSearchCV to find the best parameters for the Decision Tree Classifier. However, the F1 score we got from the model with the best parameters is lower than the default model. So we then decided to simply go with the default model.

1. Model Evaluation



We calculated the F1 score to be around 0.894, which is pretty good.

1. Future Prediction



We load the data from the modified future dataset and use the decision tree model to predict the ‘target’ column. We then saved the predicted ‘target’ column and the ‘id’ column to a new dataset for the final answer.

From this final project, we learned the importance of data preprocessing and handling the data imbalance. Even with the best parameters, it is hard to get a good F1 score with bad data preprocessing. How to interpret the data and find a good way to preprocess the data costs a lot of time but it is also interesting. It is very pleasing to modify the model and gradually improve the F1 score.

We have an evenly split contribution. We worked through each process together.