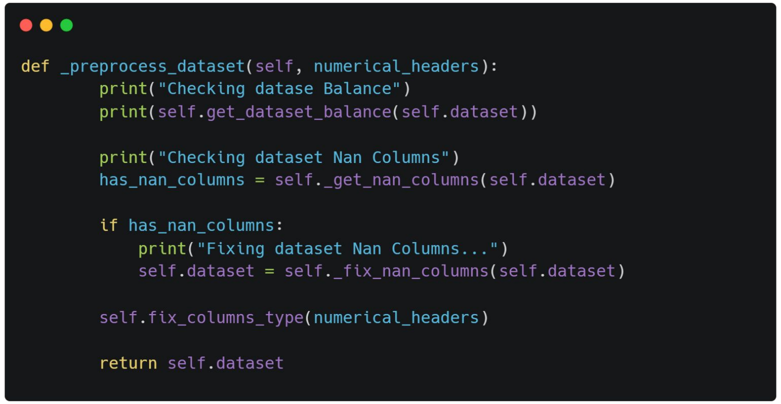
Data-Science SE

Group 56

- Bruno Moya - 56
- Joel Poma - 56
- Marc Alfonso - 56

Preprocessing



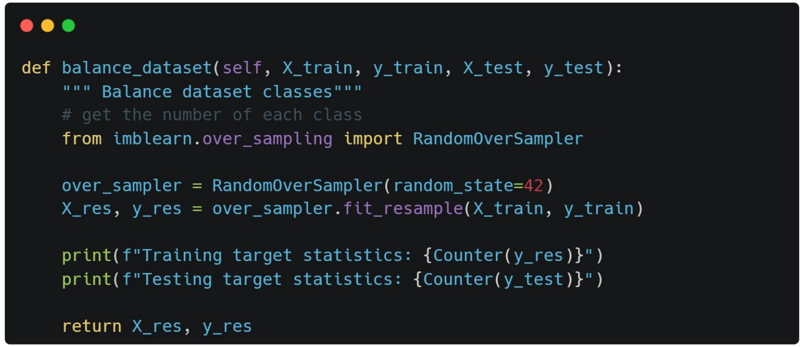
```
def _preprocess_dataset(self, numerical_headers):
        print("Checking datase Balance")
        print(self.get_dataset_balance(self.dataset))

        print("Checking dataset Nan Columns")
        has_nan_columns = self._get_nan_columns(self.dataset)

        if has_nan_columns:
            print("Fixing dataset Nan Columns...")
            self.dataset = self._fix_nan_columns(self.dataset)

        self.fix_columns_type(numerical_headers)

        return self.dataset
```

First of all before training any model, preprocessing needs to be done. This assures some ‘data cleaning’ that can help us remove bias and outliers.

We check if the database is balanced, or if we have any NaN value. Next thing was the float values that are in string format, and needed to be converted as well we did some normalization on the numeric values.

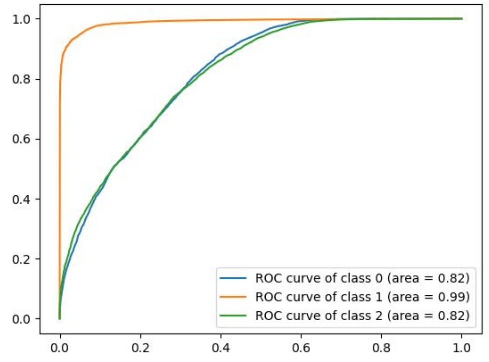
Finally, some variables were strings, and so, we could convert them to categories. This allowed us to add certain important variables into play. Apart from this categorization, handy functions like SelectKBest, allowed us to search the variables with a higher importance.

Using Apache Tika, we could process the data on the PDF train data, even though it did not help balancing our dataset, let us know there was some attributes data sources like .CSV didn’t have.

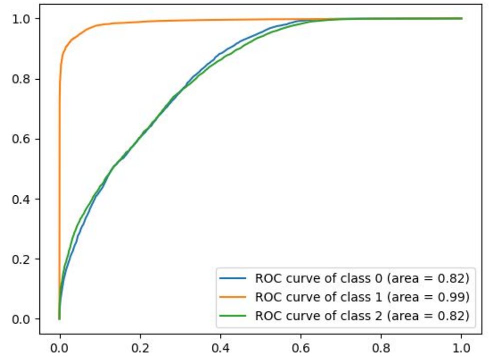


```
def balance_dataset(self, X_train, y_train, X_test, y_test):
    """ Balance dataset classes"""
    # get the number of each class
    from imblearn.over_sampling import RandomOverSampler

    over_sampler = RandomOverSampler(random_state=42)
    X_res, y_res = over_sampler.fit_resample(X_train, y_train)

    print(f"Training target statistics: {Counter(y_res)}")
    print(f"Testing target statistics: {Counter(y_test)}")

    return X_res, y_res
```

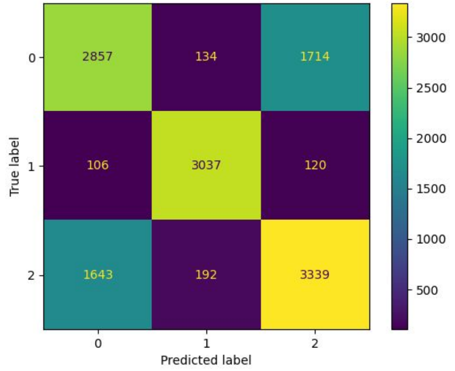
Random Forest Algorithm (We tried more models but these two slides have the best results)



ROC Curve



Recall Score

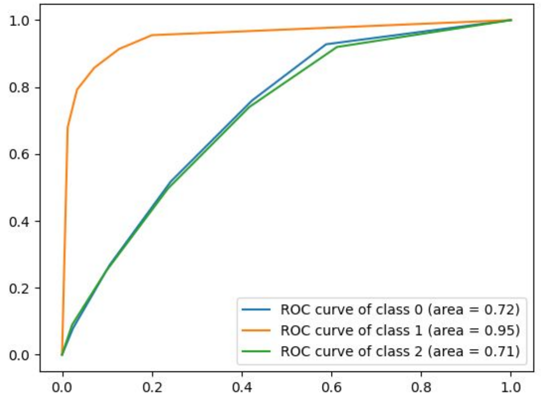


True \ Predicted	0	1	2
0	2857	134	1714
1	106	3037	120
2	1643	192	3339

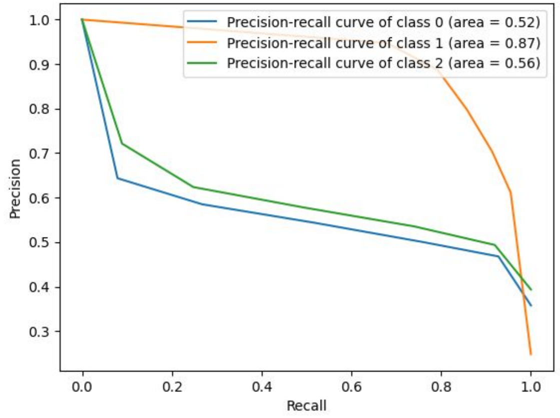
Confussion Matrix

As can be seen in the images, the area under the curve shows us our model is best at classifying class 1, having the best value overall.

KNN Algorithm



ROC Curve



Recall Score

For the KNN algorithm, the area below the ROC curve is very good for class 1, even though 0 and 1 are not that good in precision and recall. We feel something needs to be done with classes 0 and 2 as class 1 has given us best results.