Oeson
Inspiring generation

OESON PROJECT 3

Anass El Aqli

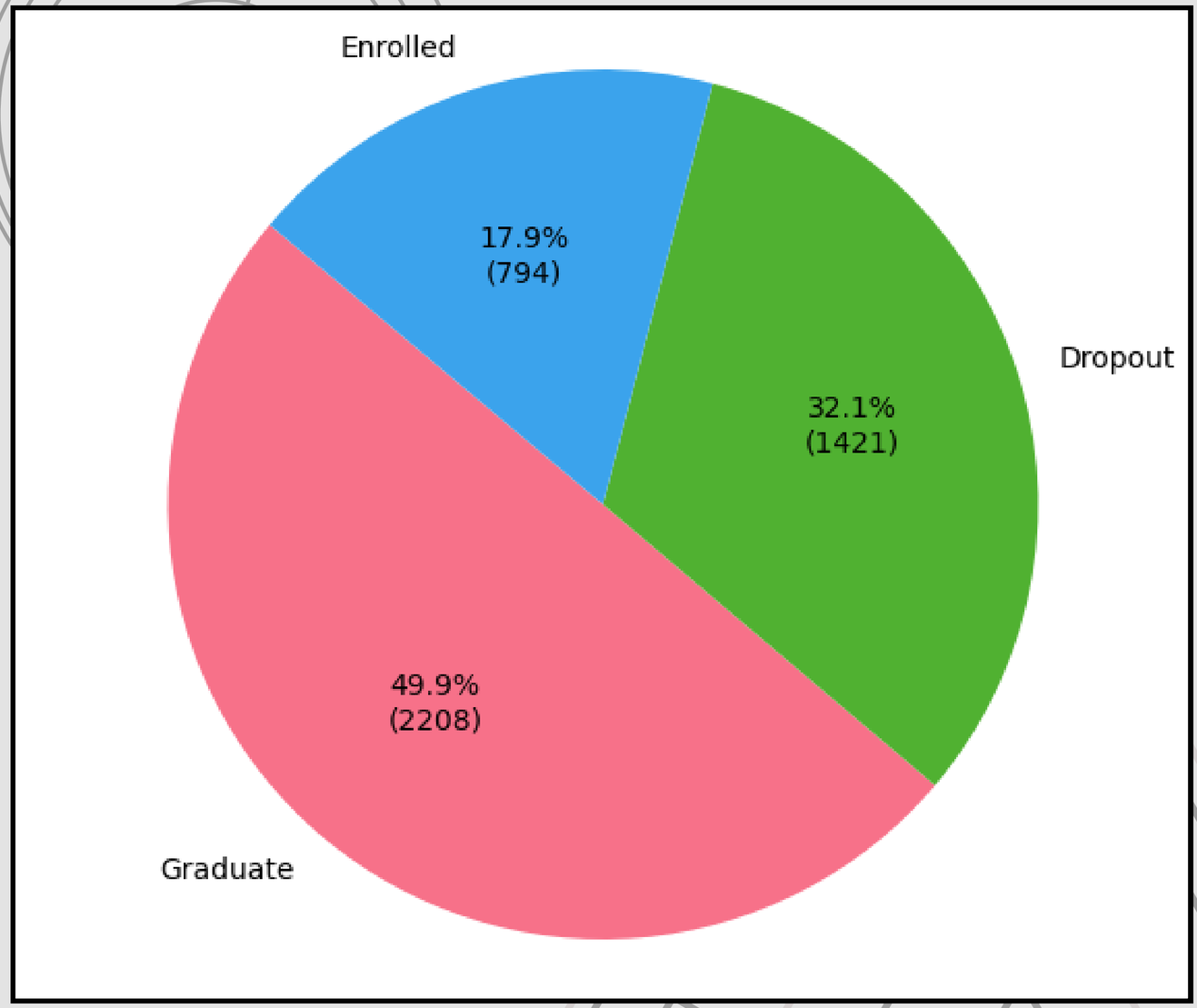
Content

- Introduction
- Visualizations
- Preparing data for ML
- Logistic Regression
- More Visualizations
- Decision Tree
- Support Vector Machine
- Random Forest
- Gaussian Naive Bayes
- Conclusion

Visualizations

Before going into machine learning We can see that our Target column consists of 3 types of values (Enrolled, Dropout, and Graduate).

Our focus for this project is whether the student will graduate or drop out. Thus, we will exclude the records with Target=’Enrolled’.

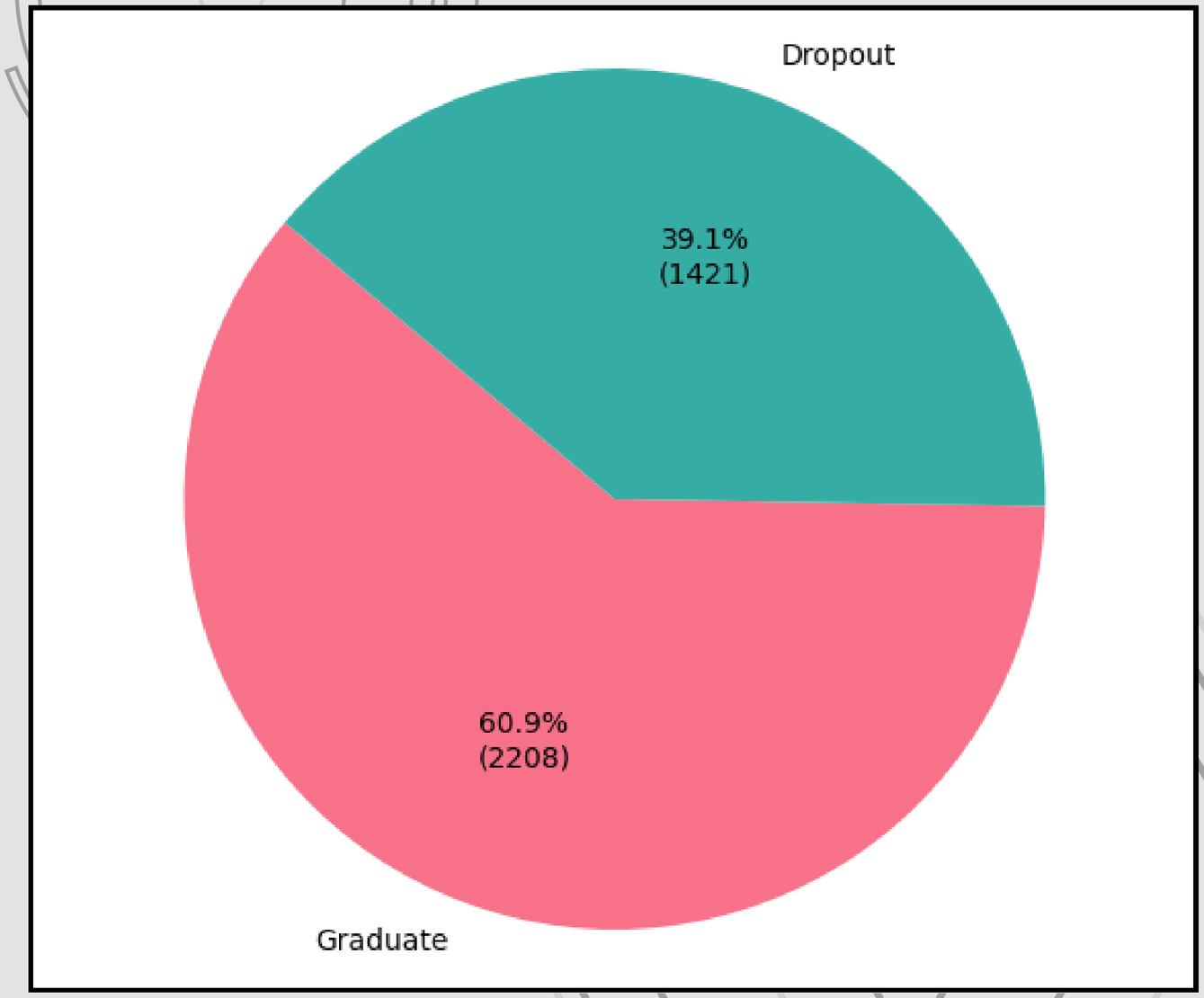


Preparing data for ML

Now, we excluded the records where Target='Enrolled'.

But still, the Target column has text values. We need to make them numerical so the ML algorithms can deal with them.

```
df = df[df['Target'] != 'Enrolled']
```



Preparing data for ML

- Using LabelEncoder from the Sklearn library we translated ‘Target’ text data to numerical data in a column we named ‘target’.
- Then for the input data that we’ll use to train our models we’ll use all the columns except ‘Target’ and ‘target’ columns.
- We’ll only keep the numerical target column for our target data.

```
[534] from sklearn.preprocessing import LabelEncoder
      le_target = LabelEncoder()
```

```
df['target'] = le_target.fit_transform(df['Target'])
```

Target	target
Dropout	0
Graduate	1
Dropout	0
Graduate	1
Graduate	1

```
244] inputs = df.drop(['Target', 'target'], axis='columns')
     target = df['target']
```

Train Test Split

- Now, we will split the data into training and testing data.
- I split it into 80% training data and 20% testing data.

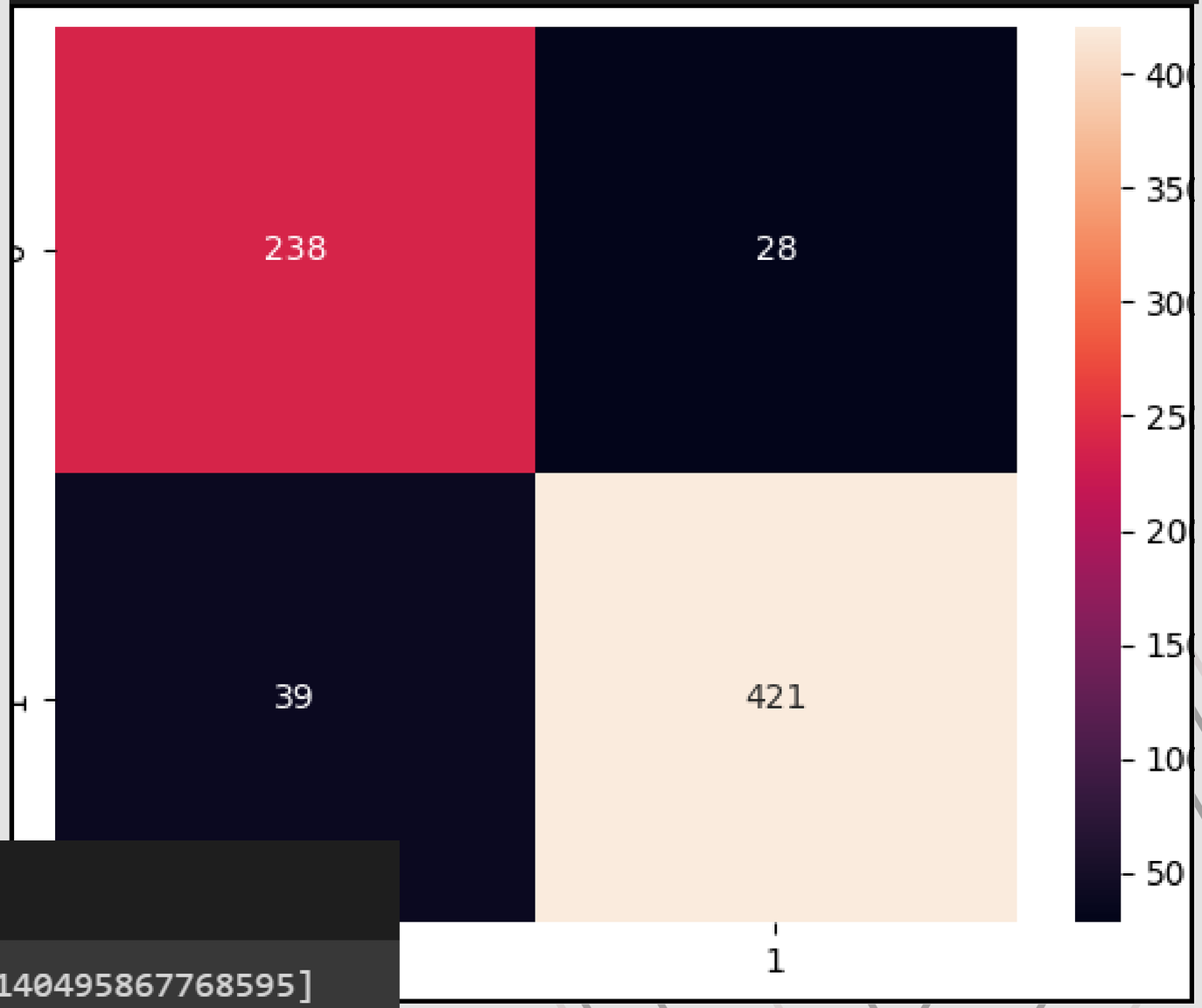
```
# train test split
from sklearn.model_selection import train_test_split
```

```
[248] X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.2, random_state = 42)
```

Logistic Regression

- So using regression, and from the confusion matrix, we can see that our model was right 238 times about the student dropping out but wrong 28 times. The model was right 421 times and wrong 39 times when it predicted that someone would graduate.
- Using K Fold Cross Validation we can see that the score of our model performance is somewhere around those values:

```
from sklearn.linear_model import LogisticRegression
logReg = LogisticRegression()
```

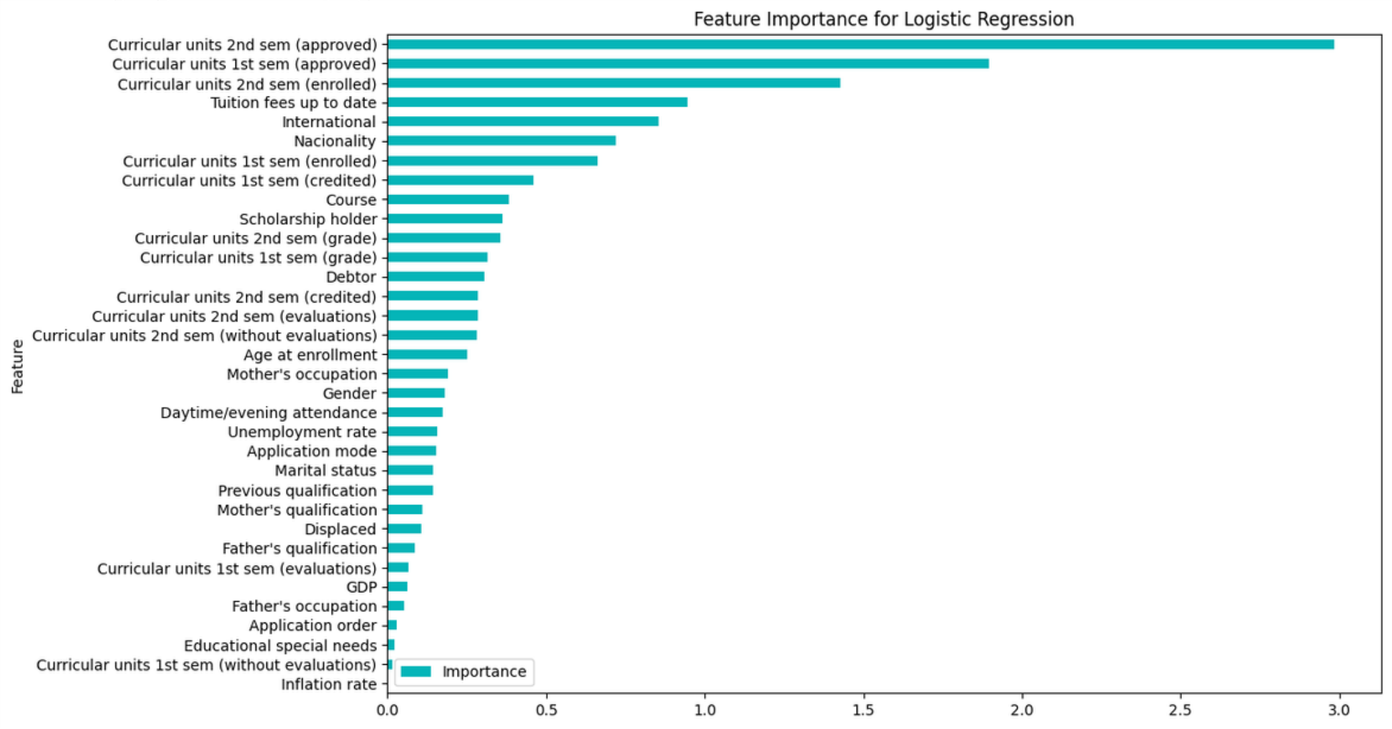


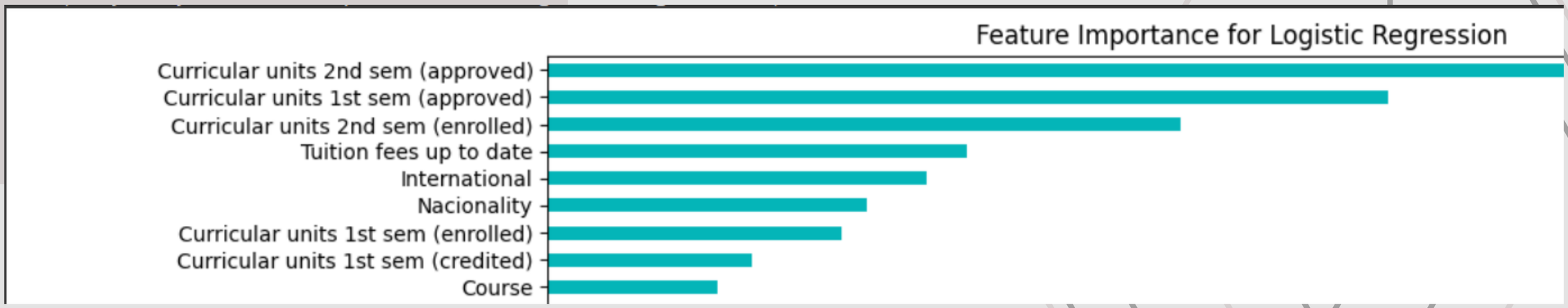

```
501] scores_logistic
```

```
[0.9165289256198347, 0.9074380165289256, 0.9140495867768595]
```

Logistic Regression

- These are the columns by their importance for the Logistic Regression.
- As it’s demonstrated, Curricular units 2nd sem (approved) played the biggest role.
- Let’s visualize some of these features in relation to graduating or dropping out.

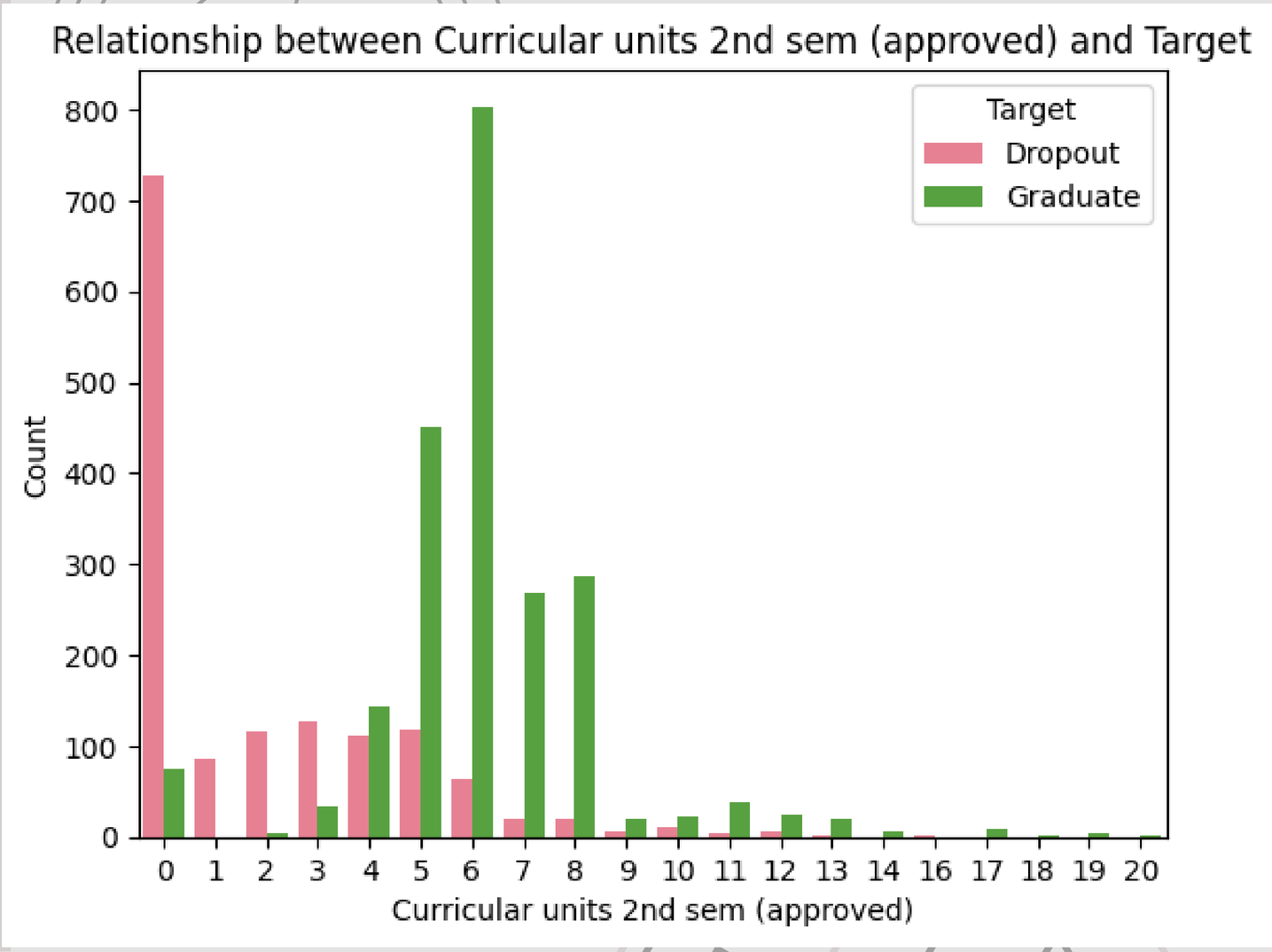




Visualizations

- We can see Curricular units 2nd sem (approved) indeed have lots of influence on dropping out or graduating.

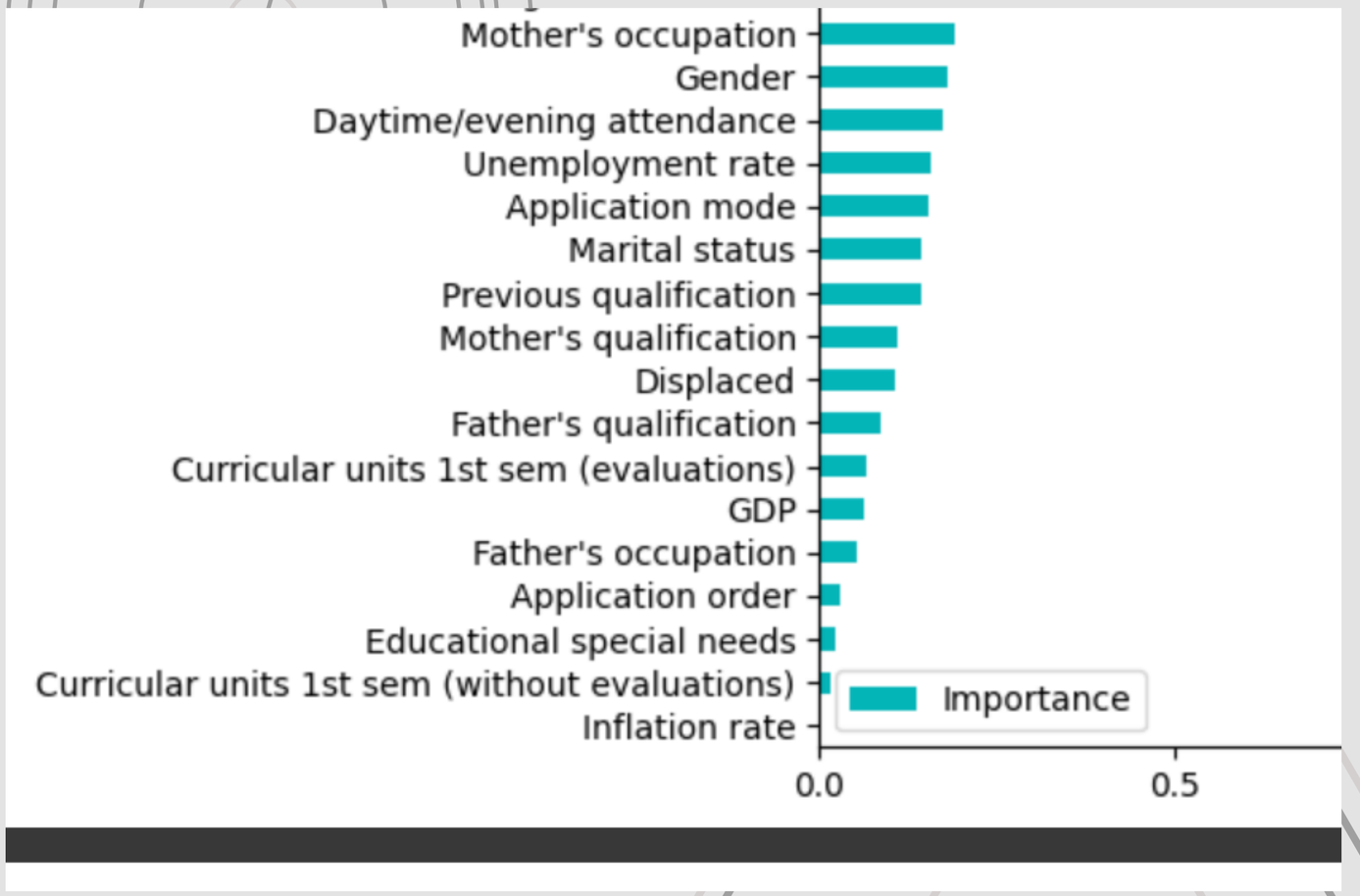
- For example, the majority of those with curricular unit number 0 dropped out, while most of those with curricular unit 6 graduated.



Visualizations

- These were the least important features for our regression.

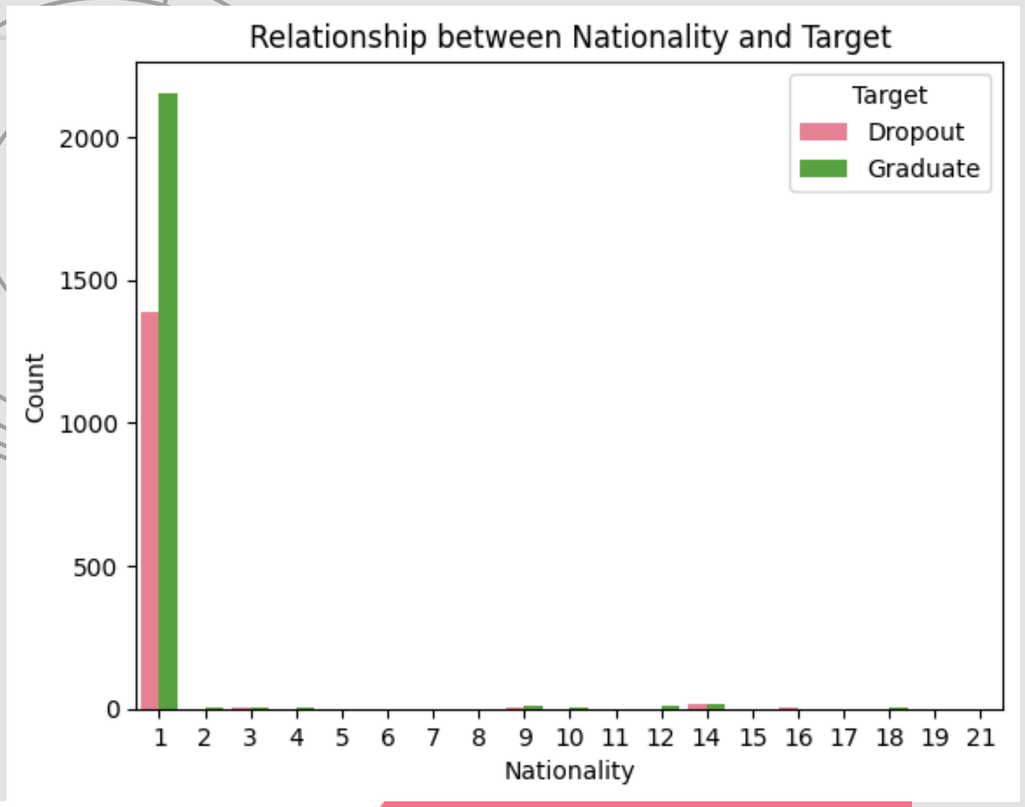
- It is nice to know that factors like educational special needs did not play a huge role in deciding whether someone graduate or not.

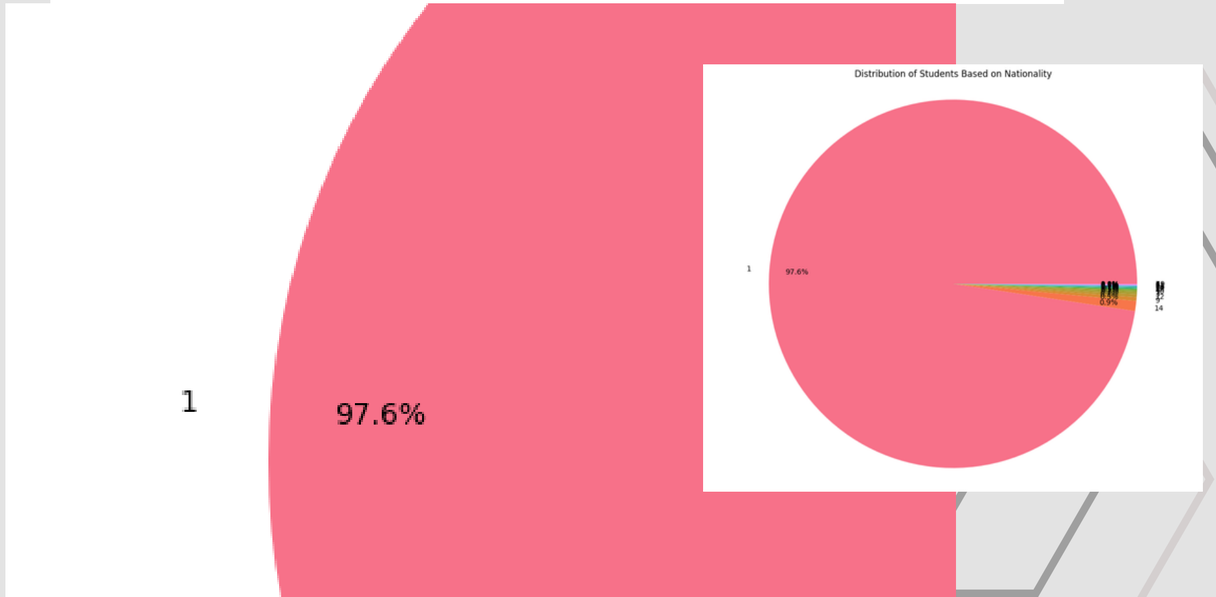


Visualizations

- For features like nationality we can barely see how influential it is just by plotting it because 97,6% of the students are from nationality number 1 which corresponds to Portuguese.

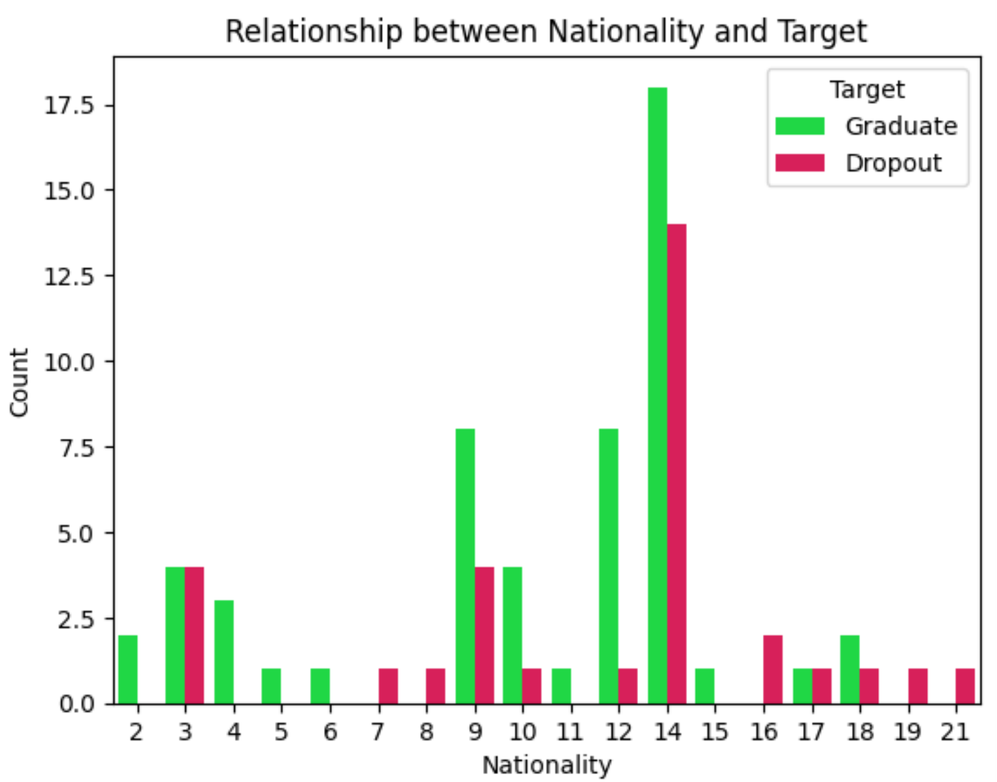
- But the good thing is more Portuguese students graduated than otherwise.

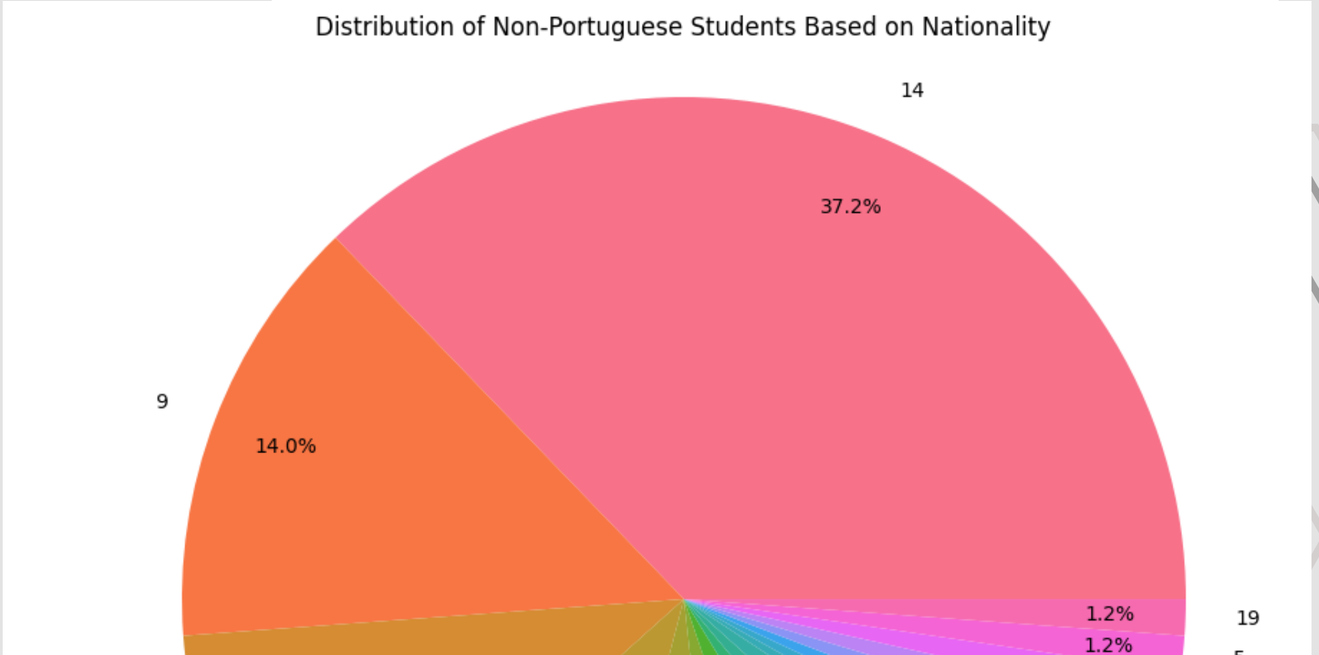




Visualizations

- When excluding Portuguese students the remaining 2.4% is made up of students from mostly other Lusophone countries (nr 14 is Brazil and nr 9 is Cape Verde).
- The influence of nationality on student status is unclear, especially considering the data is so small for nationalities that are not Portuguese.
- Since there are more than 30 columns we can’t visualize the influence of all of them in this presentation so let’s move on to other ML algorithms.



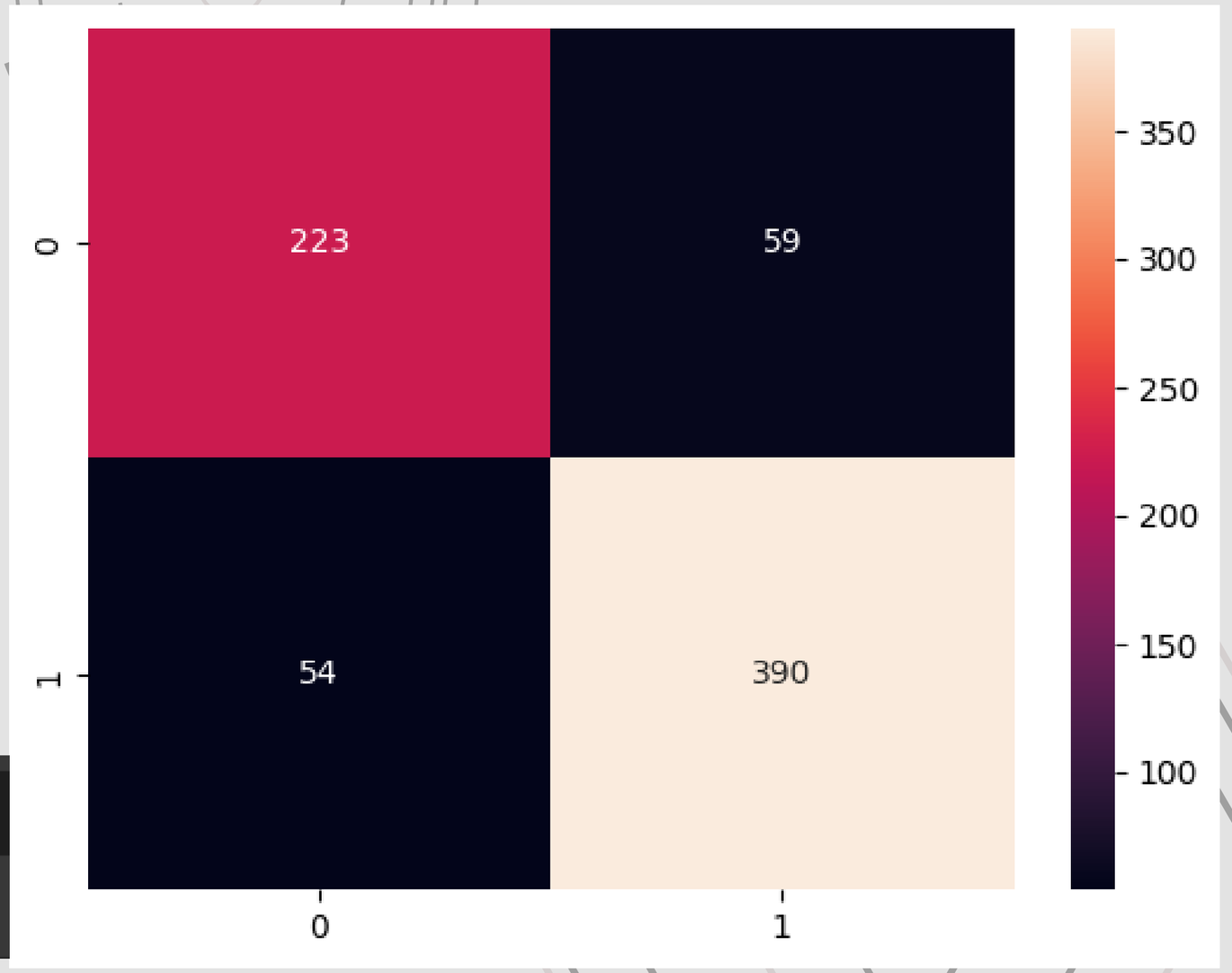


Decision Tree

- Using Decision tree algorithm, from the confusion matrix, we can see how the model performed.

- Using K Fold Cross Validation we can see that the score of our model performance is somewhere around those values which we tell it's a lot less than what we got from regression:

```
from sklearn import tree
dt_model = tree.DecisionTreeClassifier()
```




```
] scores_dt
```

```
[0.8578512396694215, 0.8413223140495868, 0.8421487603305785]
```

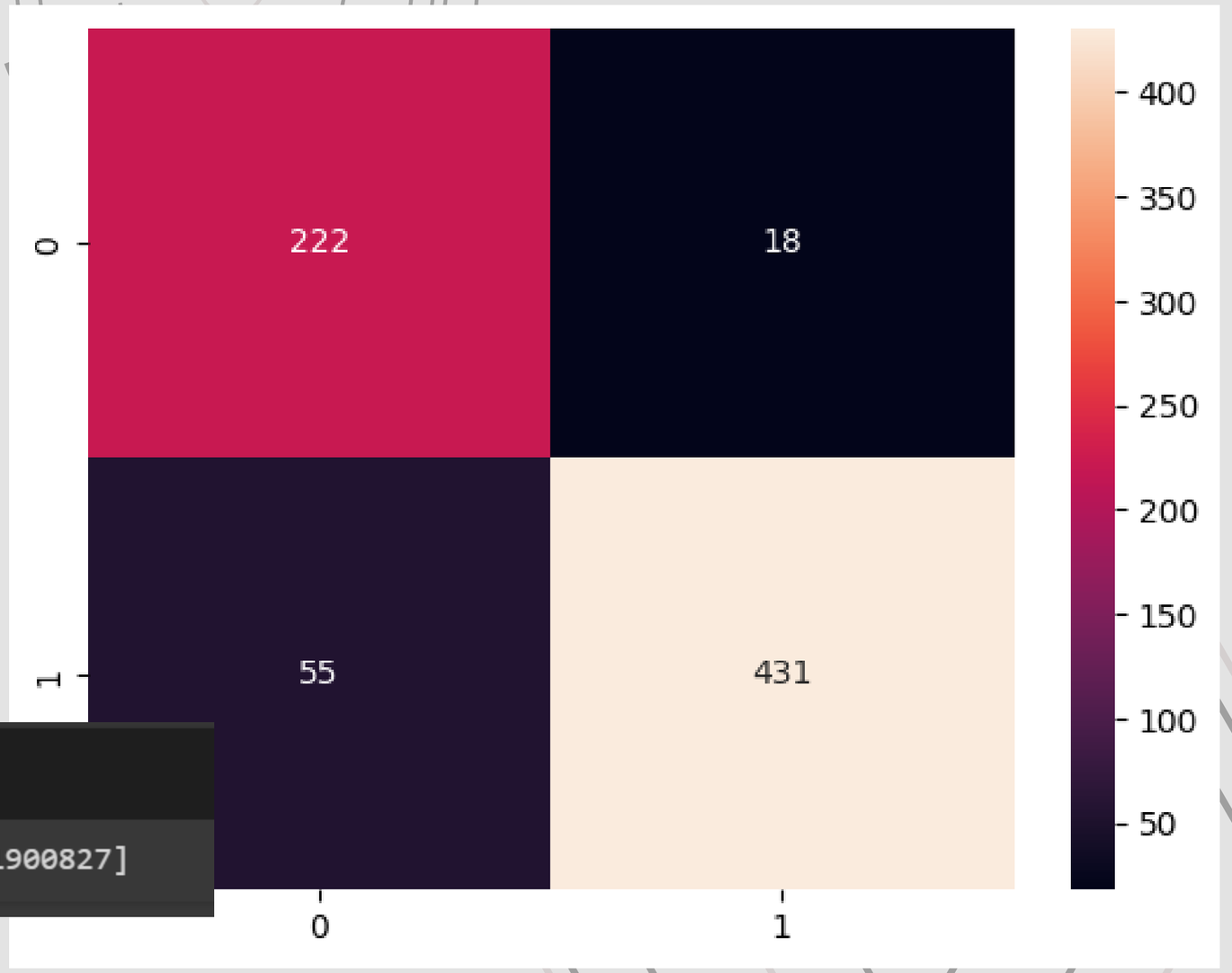
Support Vector Machine

- Using SVM, from the confusion matrix, we can see how the model performed.

- Using K Fold Cross Validation we can see that the score of our model performance is somewhere around those values:

```
[589] from sklearn.svm import SVC
      svc_model = SVC()
```

0s




```
scores_svm
```

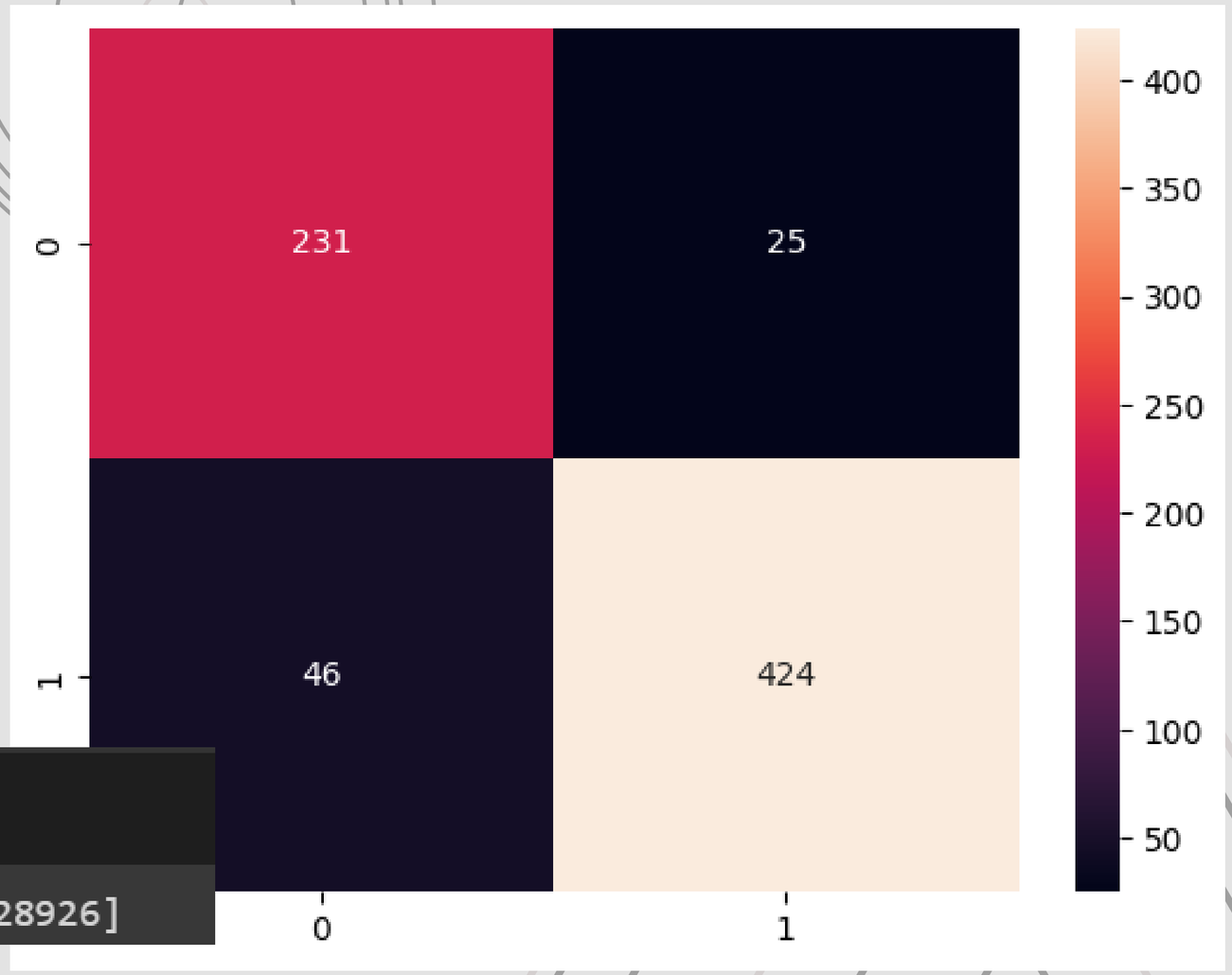
```
[0.8834710743801653, 0.8768595041322315, 0.8917355371900827]
```

Random Forest

- Using random forest algorithm, from the confusion matrix, we can see how our model performed with n_estimators=50.

- Using K Fold Cross Validation we can see that the score of our model performance is somewhere around those values:

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=50)
```




```
scores_rf
```

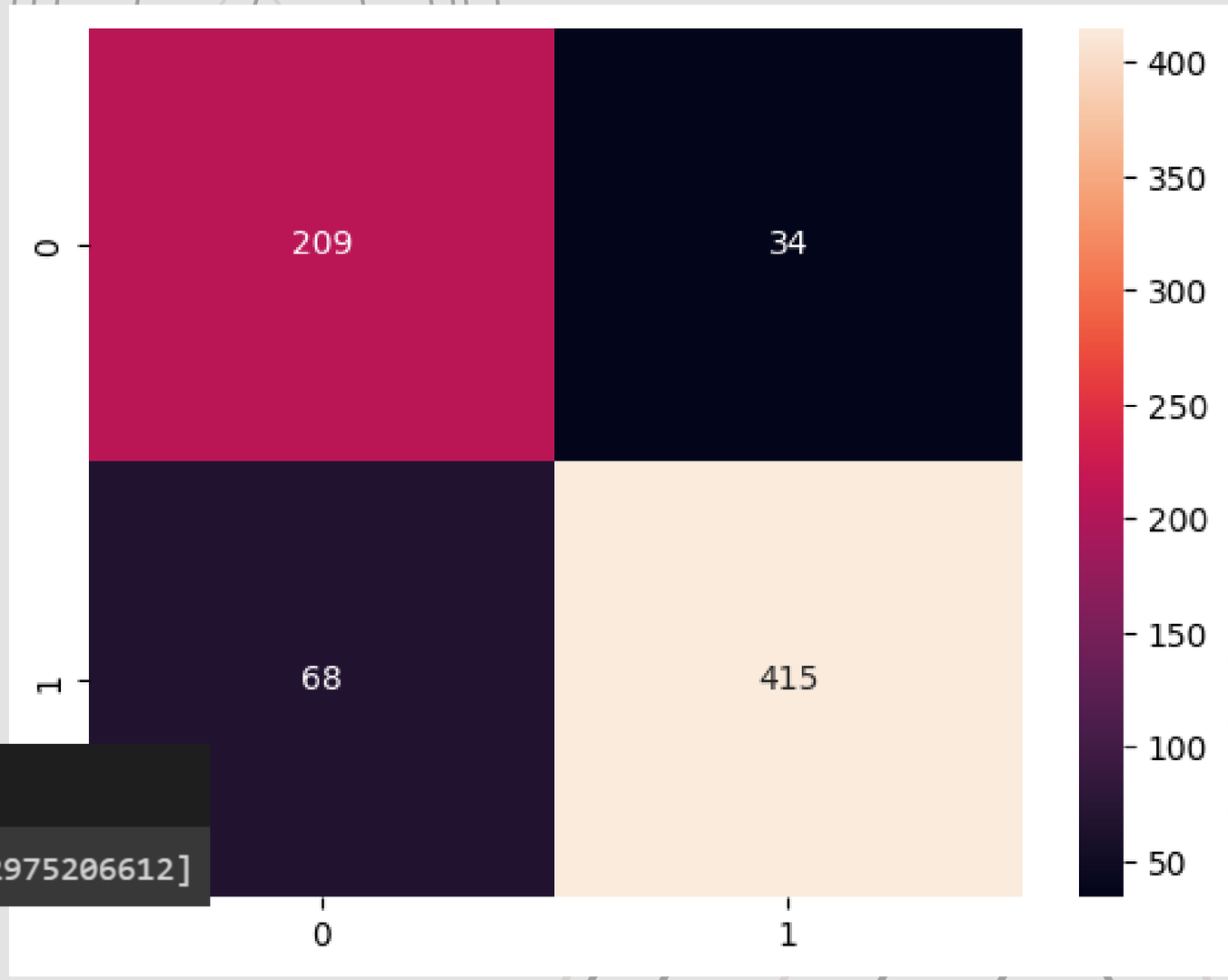
```
[0.9049586776859504, 0.9008264462809917, 0.9107438016528926]
```

Gaussian Naive Bayes

- From the confusion matrix, we can see how our model performed.

- Using K Fold Cross Validation we can see that the score of our model performance is somewhere around those values:

```
from sklearn.naive_bayes import GaussianNB, MultinomialNB
GNB_model = GaussianNB()
```




```
scores_GNB
```

```
[0.8512396694214877, 0.8388429752066116, 0.8338842975206612]
```

Hyper parameter Tuning (GridSearchCV)

	model	best_score	best_params
0	svm	0.911846	{'C': 20, 'kernel': 'linear'}
1	random_forest	0.898898	{'n_estimators': 10}
2	logistic_regression	0.912948	{'C': 1}
3	gaussian_naive_bayes	0.844628	{}
4	decision_tree	0.860606	{'criterion': 'entropy'}

Conclusion

In conclusion, for the best predictions in our data data, Logistic regression, Support Vector Machine, and the random forest algorithms worked the best.


THANKYOU