
Introduction to Data Science Project

“Churn_data” dataset

Professors: Alípio Jorge, Pedro Ferreira

Group C: Diogo Pedrosa (202006947)

Mariana Pinto (202308355)

Pedro Pedrosa (201907366)

Table of contents

1

Abstract

2

Business
understanding

3

Data
understanding

4

Data
Preparation

5

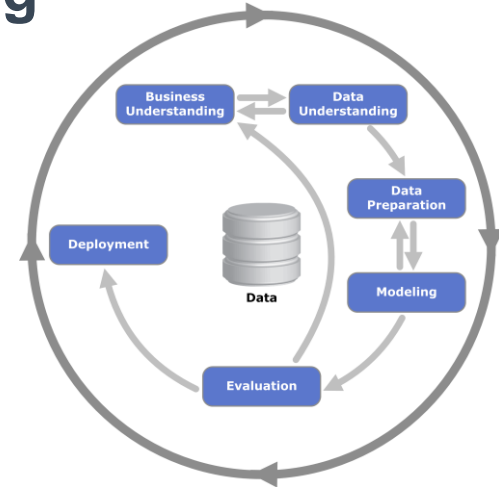
Modelling

6

Evaluation

7

Conclusion



CRISP-DM phases

Abstract

The “Churn_data” dataset:

- ✓ information about a telecom client's information
- ✓ 5000 entries with 18 different variables.

Goal:

- Predict if a client will churn or not (following the CRISP-DM methodology).

How?

1. By visualizing and understanding the data;
2. Data treatment and transformation;
3. Optimizations of the data to be applied to different models;
4. A qualitative and quantitative analysis to evaluate the performance of each model;
5. Decide which is the best model.





Business understanding

Business problem: churn of people

Business success criteria: decrease the churn of people (applying campaigns and other retention strategies having the potential of predicting churn)

Machine learning goal




Objective: predict if a client will churn or not, given the client's attributes, with high accuracy, prioritizing high recall → applying a variety of different models and evaluating them quantitatively and qualitatively so we can find a methodology that can give us the most accurate predictions possible.

Data Understanding- Attributes characterization

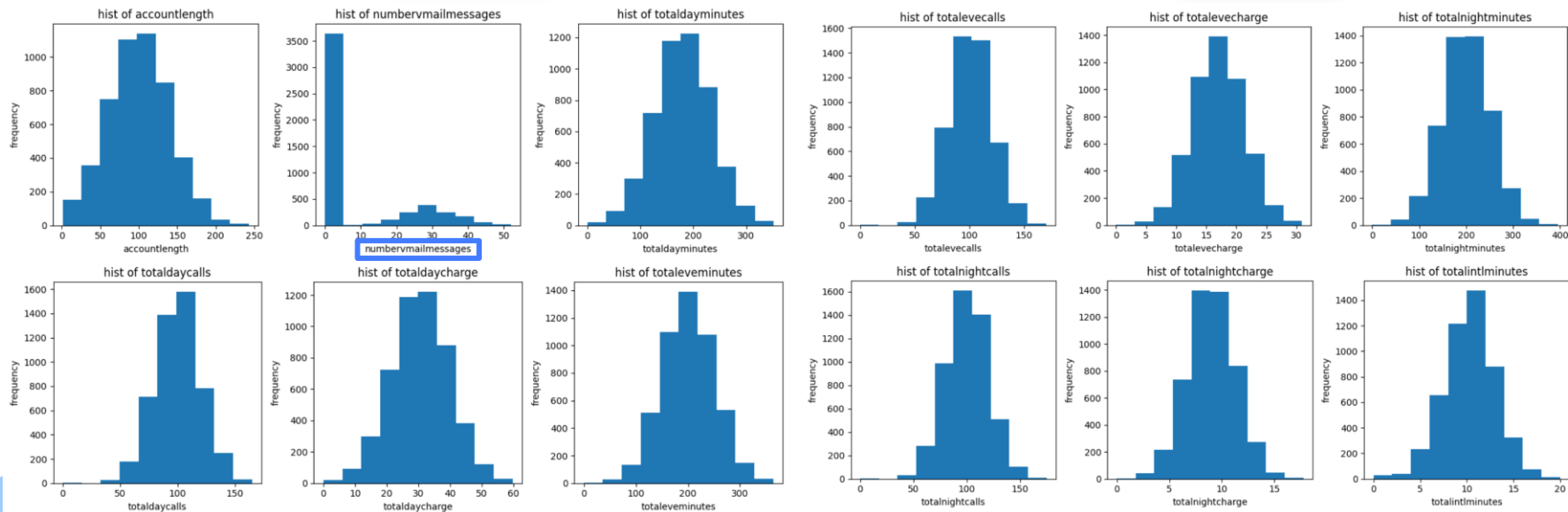
	<u>churn</u>	<u>accountlength</u>	<u>internationalplan</u>	<u>voicemailplan</u>	<u>numbervmailmessages</u>	<u>totaldayminutes</u>	<u>totaldaycalls</u>	<u>totaldaycharge</u>	<u>totaleveminutes</u>	<u>totalevecalls</u>	<u>totalevecharge</u>
0	No	128.0	no	yes	25.0	265.1	110.0	45.07	197.4	99.0	16.78
1	No	107.0	no	yes	26.0	161.6	123.0	27.47	195.5	103.0	16.62
2	No	137.0	no	no	0.0	243.4	NaN	41.38	121.2	110.0	10.30
3	No	84.0	yes	no	0.0	299.4	71.0	50.90	61.9	88.0	5.26
4	No	75.0	yes	no	0.0	166.7	113.0	28.34	148.3	122.0	12.61
5	No	118.0	yes	no	0.0	223.4	98.0	37.98	220.6	101.0	18.75
6	No	121.0	no	yes	24.0	218.2	88.0	37.09	348.5	108.0	29.62

<u>totalnightminutes</u>	<u>totalnightcalls</u>	<u>totalnightcharge</u>	<u>totalintlminutes</u>	<u>totalintlcalls</u>	<u>totalintlcharge</u>	<u>numbercustomerservicecalls</u>
244.7	91.0	11.01	10.0	3.0	2.70	1.0
254.4	103.0	11.45	13.7	3.0	3.70	1.0
162.6	104.0	7.32	12.2	5.0	3.29	0.0
196.9	89.0	8.86	6.6	7.0	1.78	2.0
186.9	121.0	8.41	10.1	3.0	2.73	3.0
203.9	118.0	9.18	6.3	6.0	1.70	0.0
212.6	118.0	9.57	7.5	7.0	2.03	3.0

Legend:

-  Categorical attributes (binary)
-  Discrete numerical attributes
-  Continuous numerical attributes

Data Understanding- Data visualization

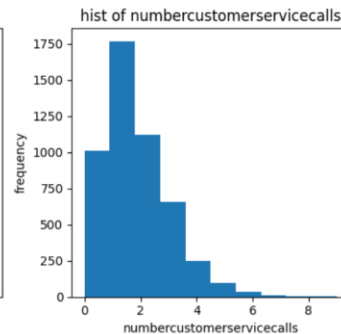
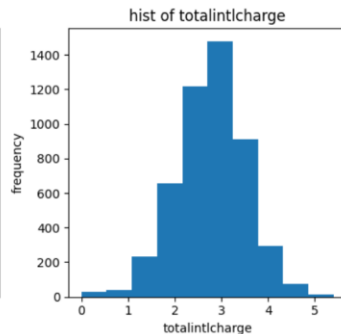
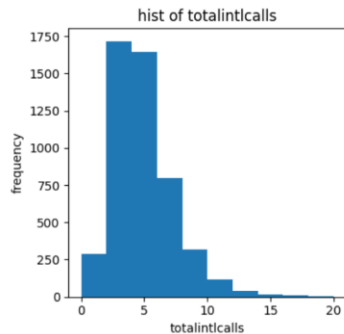


For the **numerical variables**:

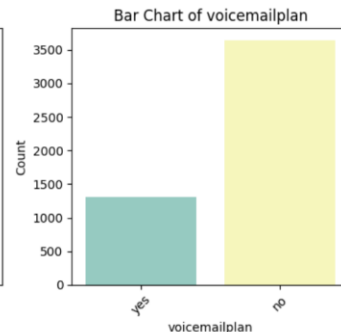
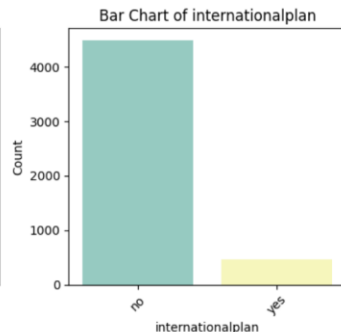
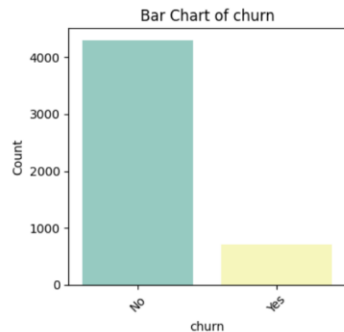
✓ they all seem to follow a normal/lognormal distribution

Except **"numbervmailmessages"** where data is not balanced (it is very biased to a specific value → 0)

Data Understanding- Data visualization

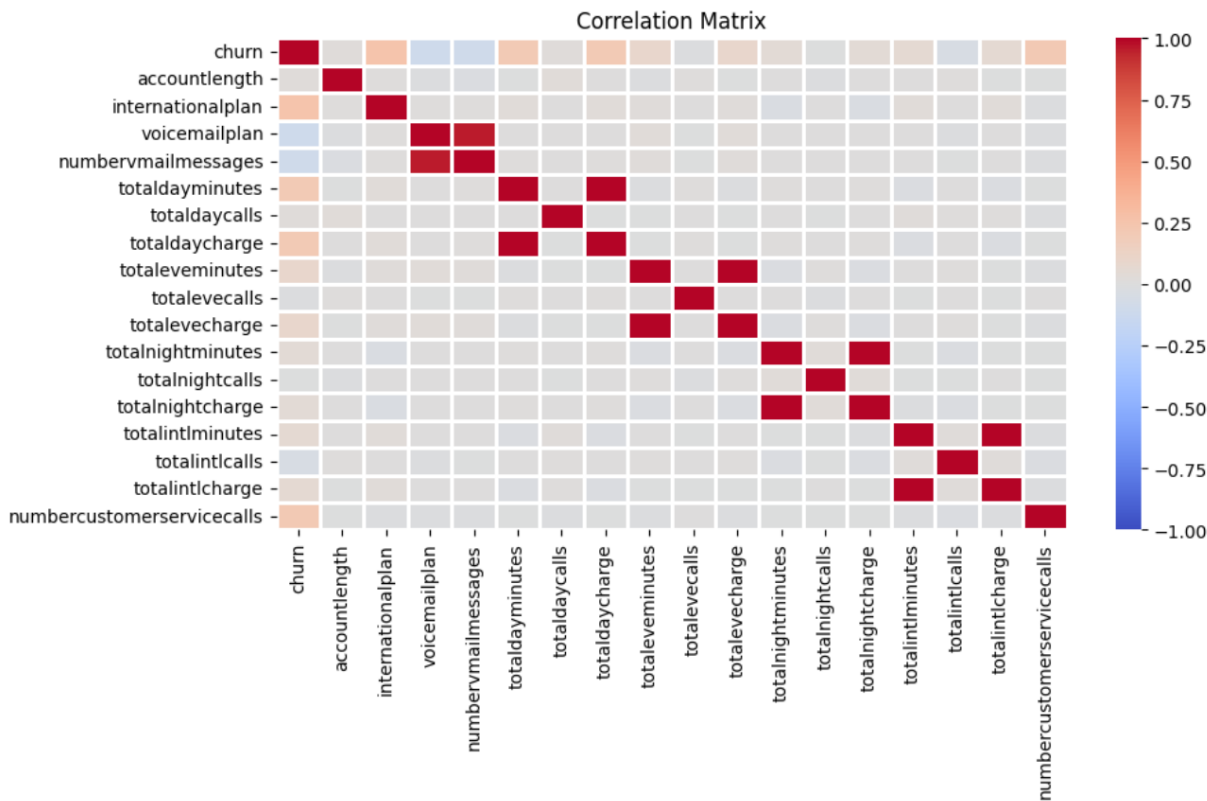


For the **numerical variables**:
same as previously (normal/
lognormal distribution).



For the **categorical variables**: data is not
balanced

Data Understanding- Correlation matrix



✓ The **total number of minutes spent on calls** on a given period of the day **and the total amount of money that was paid** for that period **are correlated**.

✓ The **number of mail messages** has a **high correlation** with the presence of a mail plan.

✓ The **international plan, total minutes during the day and the charge, and the number of customer service calls** are **correlated to the “churn” class**.

Data Preparation- Null values

```
print(f'Total number of costumers with Null parameters: {df.isna().sum().sum()}')
```

Total number of costumers with Null parameters: 849

Removing them? → No 



Data imputation

(only for clients that have one or less missing values)



Based on linear regression

```
print("Number of missing values =",df_filled.isna().sum().sum())
```

Number of missing values = 501

Data Preparation- outliers

Total number of customers with 2 or more outlier attributes: 529



Discard the clients that have 2 or more outliers

Total number of customers with 1 outlier attribute: 874



Data imputation

Based on linear regression

Data Preparation- Training and test sets

Two different sets of data:

1. Unbalanced data

- ✓ Original dataset after pre-processing
- ✓ Divided into training set and test set in **80%/20%** proportions

2. Balanced data

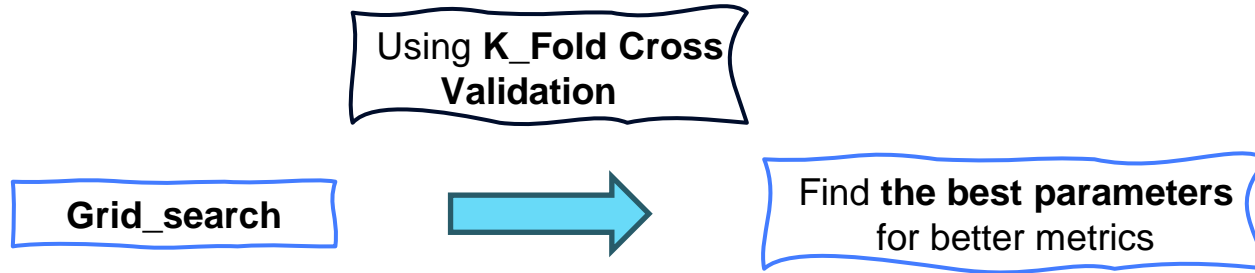
- ✓ Random selection of no churn to match churning people
- ✓ Divided into training set and test set in **80%/20%** proportions

Metrics that will be used to test the models

- **Accuracy** → For overall performance check
- **Recall** → The most important metric in our problem: the proportion of well-classified churning people with the total number of churning people.
- **Precision** → Proportion of well-classified churning people to all churning people classifications

*Recall and precision instead of F1 for better interpretability

Modelling - Methodology



Confusion matrix plot (and others) for better visualization

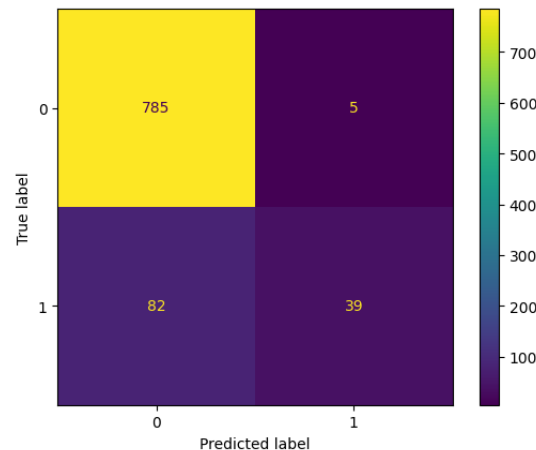
Modelling- kNN (k-Nearest Neighbours)- Unbalanced

Best parameters found using `GridSearchCV`

Fitting 10 folds for each of 72 candidates, totalling 720 fits
{'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'uniform'}

```
accuracy = 0.9018223443223444  
precision = 0.9344369303465415  
recall = 0.3188446624853477
```

- **90% accuracy looks promising** (BUT compared to the 86% of random guessers it's only a little better)
- **32% recall is awful**: about $\frac{2}{3}$ of the client churns (we are predicting they don't churn which is far from what we want → explained by the unbalanced fact)

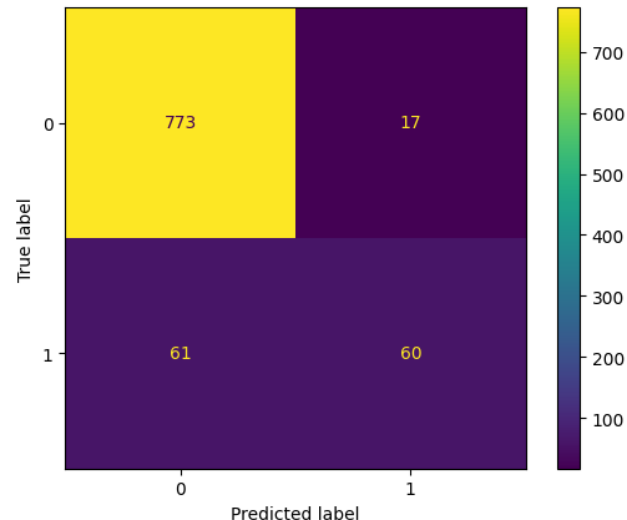


Modelling- Naive Bayes- Unbalanced

```
accuracy = 0.9079670329670331  
precision = 0.7645971668779253  
recall = 0.49022501478500746
```

- **accuracy \approx KNN accuracy (90%)**
- **17% increase in the recall and a decrease of 17% in precision** HOWEVER these metrics are still quite far from a satisfying result:

Problems: majority class is biased + assumption of class independence (which is far from being satisfied here)



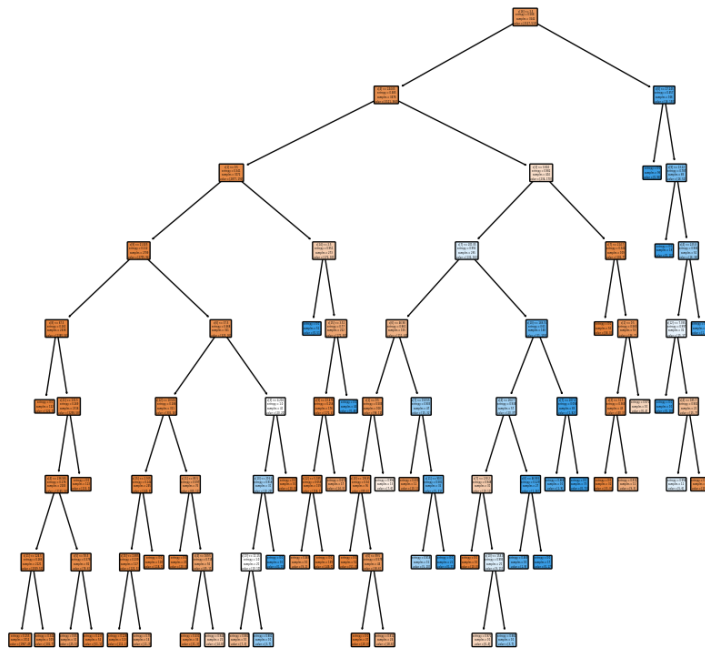
Modelling- Decision Tree- Unbalanced

Best parameters found using `GridSearchCV`

```
ccp_alpha = 0.0001, criterion= 'entropy', max_depth = 8 , min_samples_leaf= 10 , min_samples_split= 2, random_state= 42, splitter= 'best' )
```

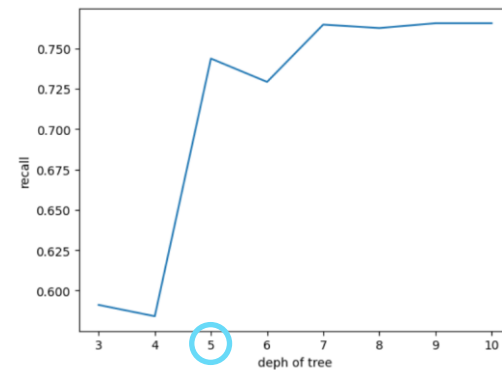
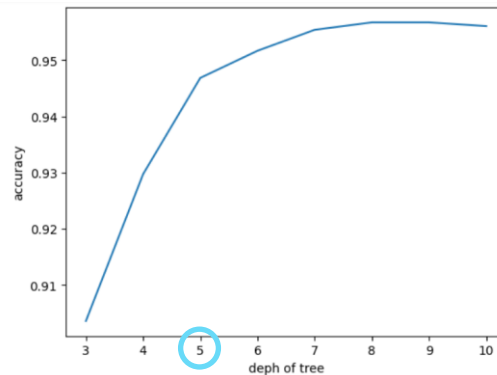
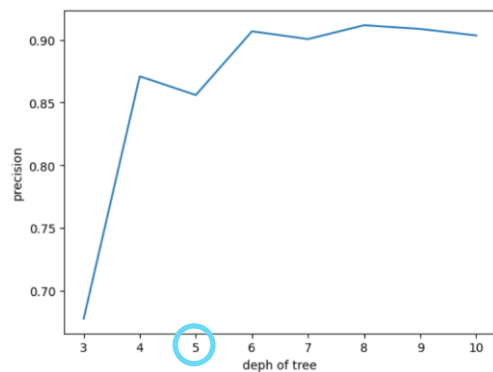
accuracy = 0.956730287256603
precision = 0.9116708392976065
recall = 0.7625034657358922

- Overall the metrics look good
- BUT the decision tree looks complex and may be overfitted



Modelling- Decision Tree- Unbalanced

Plots obtained for the metrics in function of the max_depth



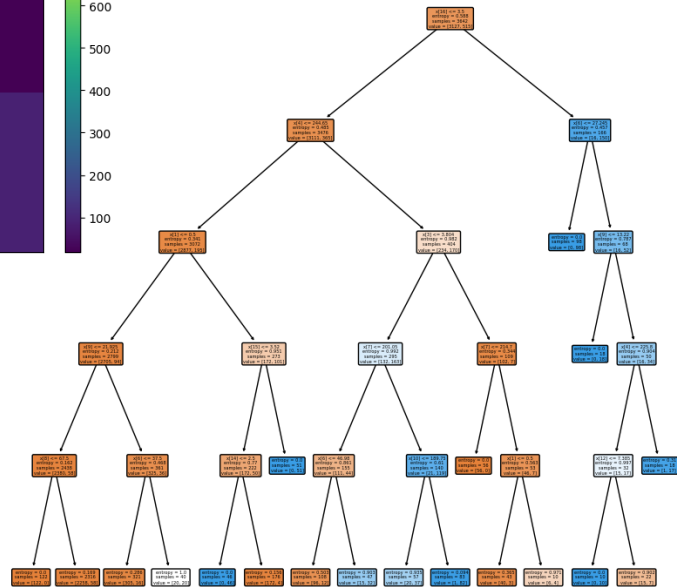
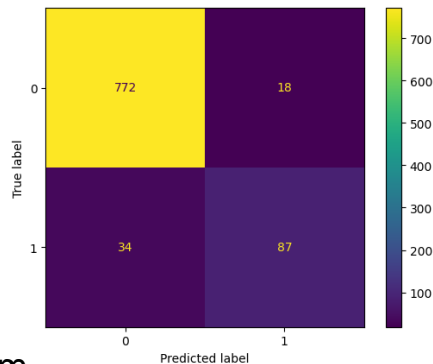
It is possible to note that the **best parameter for the depth of the tree is 5.**

Modelling- Decision Tree- Unbalanced

Knowing the max_depth = 5:

```
accuracy = 0.9468507807981492  
precision = 0.8560135351757383  
recall = 0.7436154179593657
```

- **Accuracy of 94%** (8% above a random guesser)
- **Recall is now better** (compared with the previous models): we are only mislabeling a churner 1 out of 4 times, (which isn't too bad considering the unbalanced proportion of the dataset)



Modelling- Tree Ensembles- Unbalanced

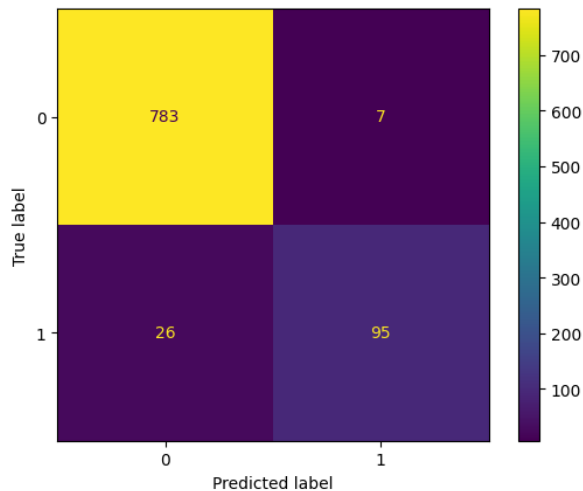
1. Random Forest

Fitting 5 folds for each of 36 candidates, totalling 180 fits

```
{'max_depth': 5, 'max_features': 8, 'n_estimators': 1000, 'random_state': 42}
```

```
accuracy = 0.9631024957479404  
precision = 0.9419839292349106  
recall = 0.782237864175527
```

- In comparison with Decision Trees, this confirms the initial statement that we'd obtain a slightly better result at the cost of losing **interpretability**



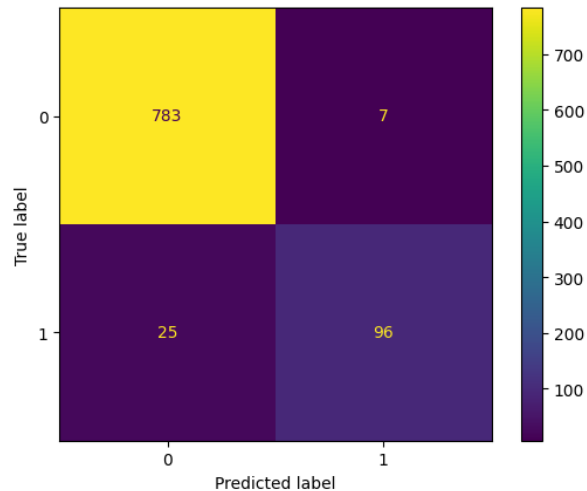
Modelling- Tree Ensembles- Unbalanced

2. Gradient Boosting

Fitting 5 folds for each of 12 candidates, totalling 60 fits
{'learning_rate': 0.05, 'n_estimators': 500, 'random_state': 42}

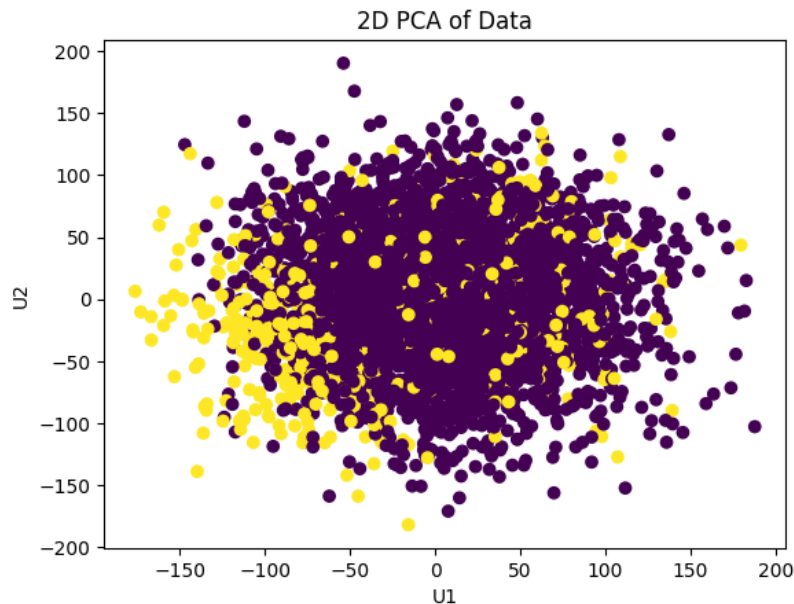
```
accuracy = 0.9633217934644938  
precision = 0.9389708000329092  
recall = 0.7869441918520144
```

- **Slight improvement over the traditional Random Forest** → this model has techniques such as e.g. regularization and better handling of unbalanced data
- **Significant improvement** relative to the original **Decision Trees**



Modelling- SVM (Support Vector Machines)- Unbalanced

1. 2-dimensional PCA decomposition of our data



Our data is far from being linearly separable



We need to consider non-linear kernels

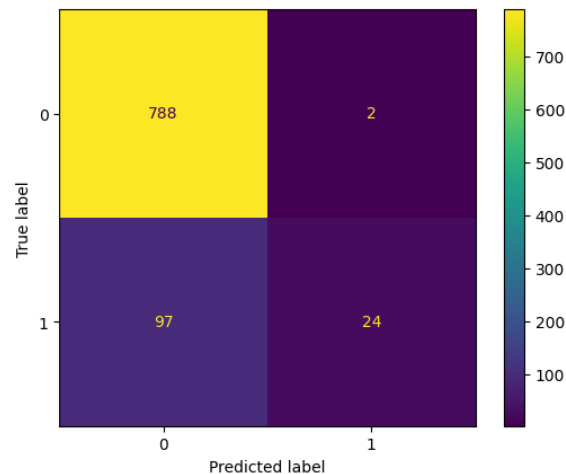
Modelling- SVM (Support Vector Machines)- Unbalanced

2. Best hyperparameters found for non-linear kernels

Fitting 5 folds for each of 15 candidates, totalling 75 fits
{ 'C': 10, 'kernel': 'rbf', 'random_state': 42 }

```
accuracy = 0.8853485482684166  
precision = 0.9262633917806333  
recall = 0.19566614232071872
```

- These results are even **worse than the first models**, but they were expected in the sense that our **data is not linearly separable**.

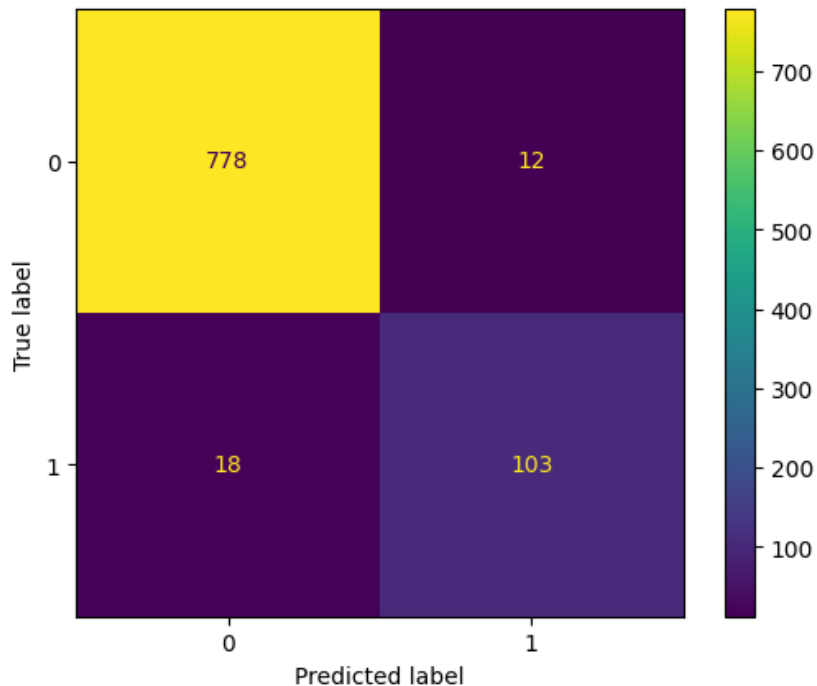


Modelling- Neural Network- Unbalanced

Fitting 5 folds for each of 45 candidates, totalling 225 fits
{'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (200, 25), 'learning_rate_init': 0.1, 'max_iter': 400, 'n_iter_no_change': 30, 'random_state': 42, 'solver': 'adam', 'verbose': False, 'warm_start': False}

```
accuracy = 0.9694720208441394  
precision = 0.943159481429557  
recall = 0.8314276148429339
```

- **Best metrics overall** until now.
- Still a **big room for improvement** with the overall possibility of NN



Modelling- kNN (k-Nearest Neighbours)- Balanced

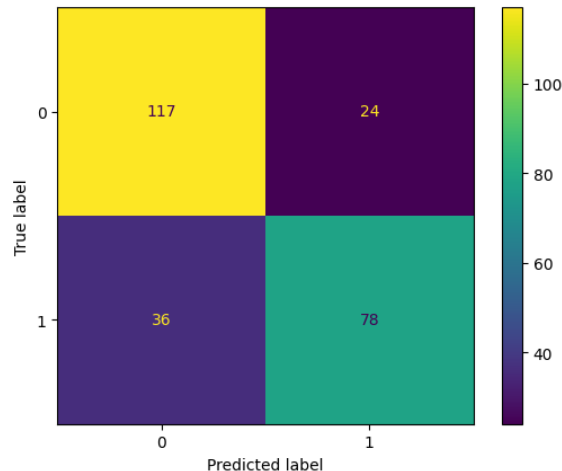
Fitting 10 folds for each of 72 candidates, totalling 720 fits
`{'metric': 'manhattan', 'n_neighbors': 10, 'weights': 'distance'}`

```
accuracy = 0.8153112696850394  
precision = 0.8813701123119715  
recall = 0.7295773754462564
```

Compared to the unbalanced dataset model:

- the **precision is similar** (a little worse but nothing substantial)
- the accuracy seems almost 10% worse (note that now the random guessing is only 50%, so **this lower accuracy means a better accuracy**)
- the **recall is much bigger**

CONCLUSION: the balanced dataset gives better results than the unbalanced one for the KNN model (also comparing the confusion matrix we can see a very big decrease in the proportion of the miss classifications of the false negatives)

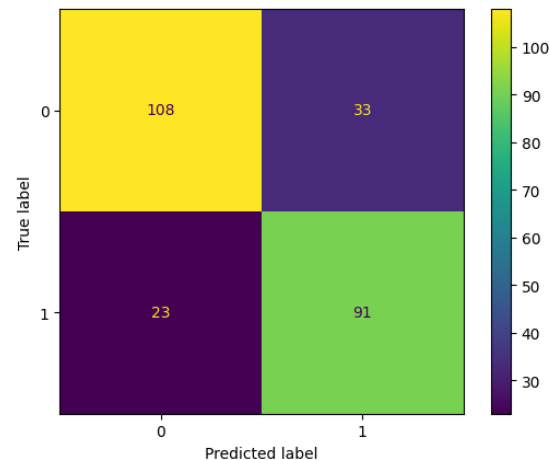


Modelling- Naive Bayes- Balanced

```
accuracy = 0.8019192913385828  
precision = 0.7907224345202949  
recall = 0.8214038766119842
```

- \approx to kNN model
- Overall **increase in the quality of the metrics** (better precision, much better recall and symbolically better accuracy).

CONCLUSION: even though this accuracy is lower it has more meaning than the previous ones, as there's no bias in the dataset. Also, by using a balanced dataset our custom scoring prioritizing recall has now a clear effect, as we encounter for the first time the recall as the superior metric. This all serves as evidence that using a balanced dataset is beneficial in this case.



Modelling- Decision Tree- Balanced

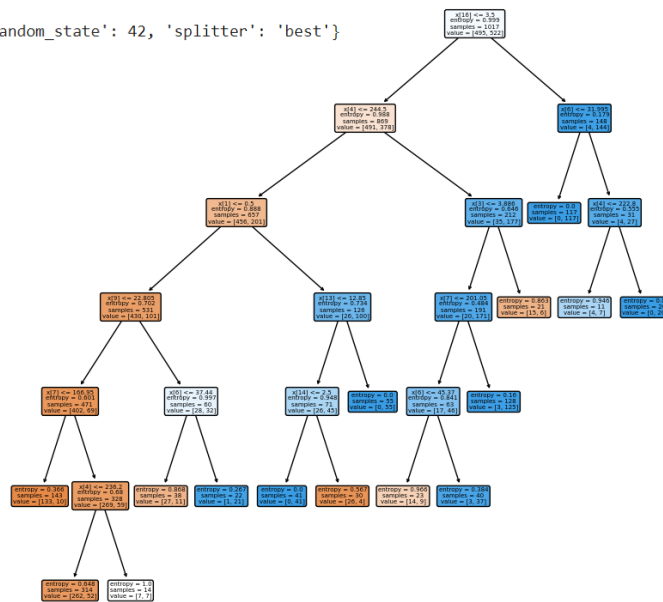
Best parameters found using GridSearchCV

Fitting 5 folds for each of 7938 candidates, totalling 39690 fits

{'ccp_alpha': 0.005, 'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 10, 'min_samples_split': 30, 'random_state': 42, 'splitter': 'best'}

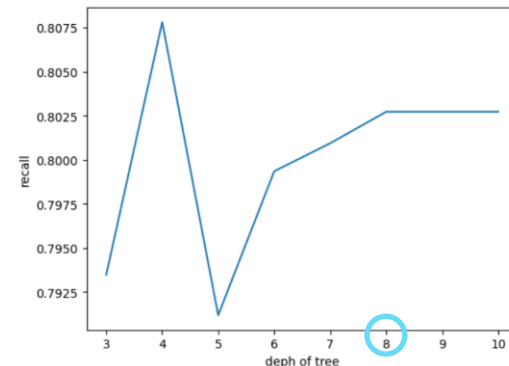
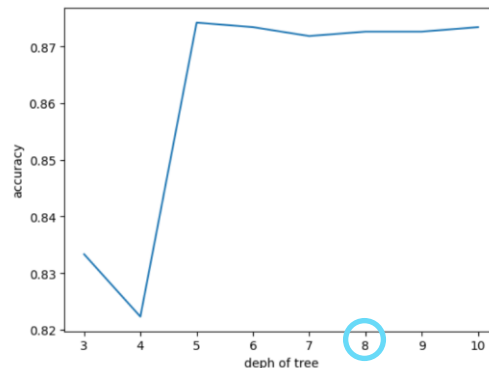
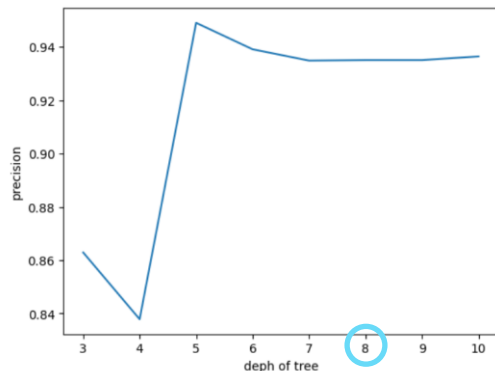
```
accuracy = 0.872613188976378
precision = 0.9350198141952706
recall = 0.8027224803863262
```

- The parameters are in general very similar to the tree using unbalanced data and the performance of the has a similar pattern to the previous models.
- An accuracy lower (in proportion means a more accurate model than before), a lower precision (but again is the least significant metric), and a little bigger recall.



Modelling- Decision Tree- Balanced

We plotted the metrics in function to the depth to see if has a better model in terms of complexity vs performance



It is possible to note that the precision and accuracy stabilized at depth 5, but the recall only later 8, hence we chose **8 as the ideal depth**.

Modelling- Decision Tree- Balanced

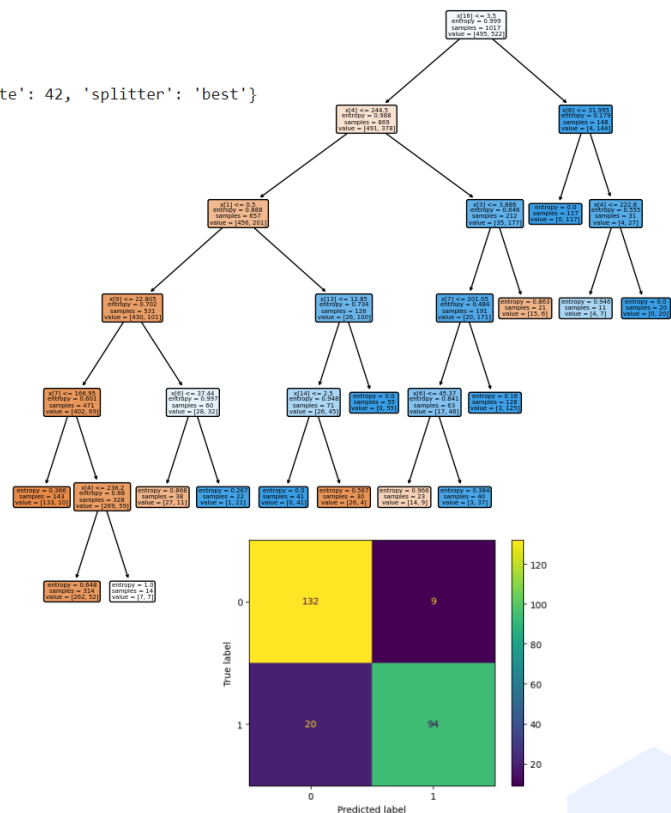
Best parameters found using GridSearchCV

Fitting 5 folds for each of 7938 candidates, totalling 39690 fits

```
{'ccp_alpha': 0.005, 'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 10, 'min_samples_split': 30, 'random_state': 42, 'splitter': 'best'}
```

```
accuracy = 0.872613188976378
precision = 0.9350198141952706
recall = 0.8027224803863262
```

- This can be interpreted in the sense that the decision tree is, to a certain degree, robust to bias in the data, which is seen here by obtaining similar metrics.



Modelling- Tree Ensembles- Balanced

1. Random Forest

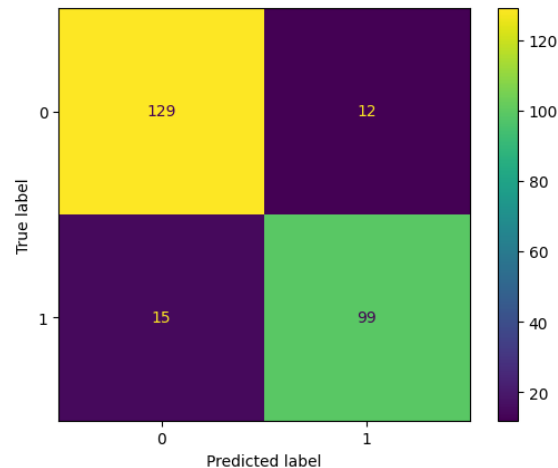
Fitting 5 folds for each of 48 candidates, totalling 240 fits

```
{'max_depth': 8, 'max_features': 8, 'n_estimators': 1000, 'random_state': 42}
```

```
accuracy = 0.8962297359888838  
precision = 0.9383239219857188  
recall = 0.84823743415839
```

- Overall performance in metrics (90% accuracy, 94% precision and 85% recall)
- Compared to the unbalanced one: a 6% decrease in accuracy, a similar precision and a significant 7% boost in the recall.

CONCLUSION: Much like before we see this model slightly outperforming the "standalone model" for a decision tree.



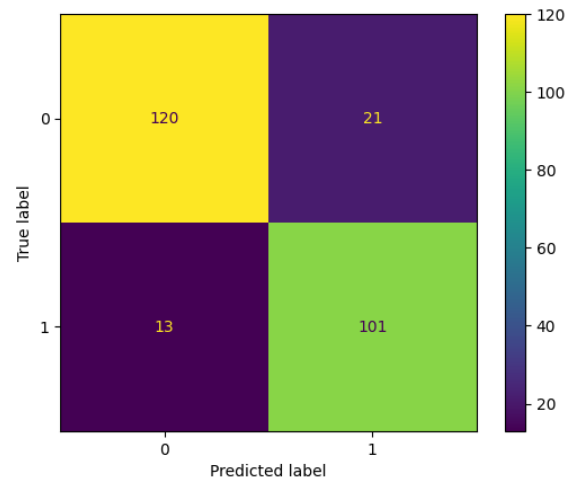
Modelling- Tree Ensembles- Balanced

2. Gradient Boosting

Fitting 5 folds for each of 12 candidates, totalling 60 fits
{'learning_rate': 0.01, 'n_estimators': 1000, 'random_state': 42}

```
accuracy = 0.8844403273120273  
precision = 0.908321329524108  
recall = 0.8570016649524584
```

- We're getting **slightly worse results in comparison to traditional ensembles** → might be associated with the fact that GBMs filter bias by themselves and require more data for the gradient boosting method (with a specific learning rate) to be able to converge.
- **Compared to the previous model** we see the usual slight decrease in accuracy accompanied by a boost in recall.



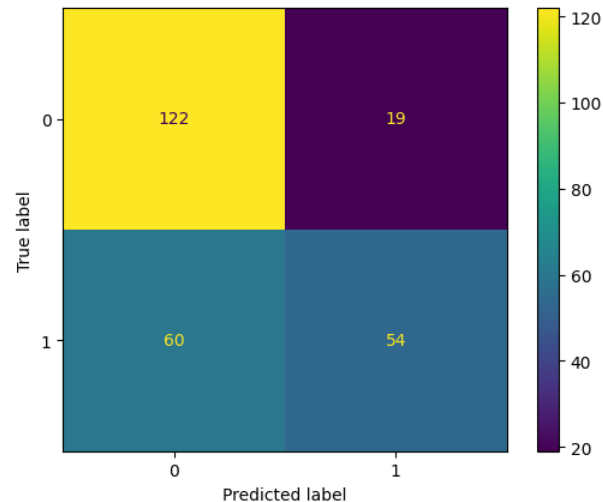
Modelling- SVM (Support Vector Machines)- Balanced

Fitting 5 folds for each of 15 candidates, totalling 75 fits
{'C': 10, 'kernel': 'rbf', 'random_state': 42}

```
accuracy = 0.6910236220472441  
precision = 0.8351035305770267  
recall = 0.4772954748124958
```

- Even though we see the typical accuracy and recall decrease/increase respectively, the **results are far from good**
- Looking at the confusion matrix → we mislabeled more people as non-churners (that were churners) than labelled correctly churners.

CONCLUSION: This model wouldn't be viable for deployment.



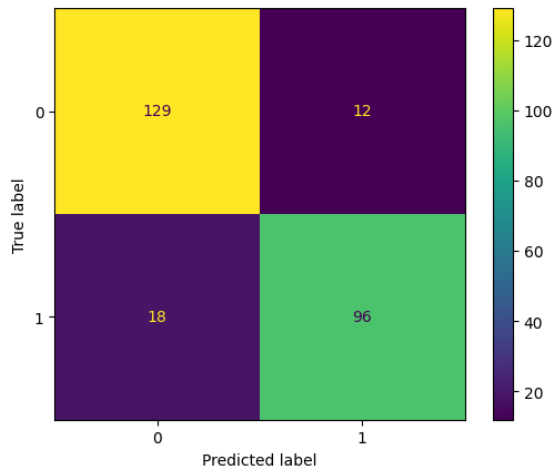
Modelling- Neural Network- Balanced

Fitting 5 folds for each of 45 candidates, totalling 225 fits

```
{'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (200, 20), 'learning_rate_init': 0.1, 'max_iter': 400, 'n_iter_no_change': 40, 'random_state': 42, 'solver': 'adam', 'verbose': False, 'warm_start': False}
```

```
accuracy = 0.8796973907673307  
precision = 0.9138489184284282  
recall = 0.8396526101690212
```

- Compared to the unbalanced dataset, it seems that there is not much difference (maybe it's even a **little worse with the balanced one**)
- It may seem strange, but we have less data in the balanced dataset, and **neural networks need a good amount of data to converge to a good result** → with this less data, but balanced data resulted in similar performance (probably with a bigger balanced set, we should have better results)



Modelling- Mean result aproach

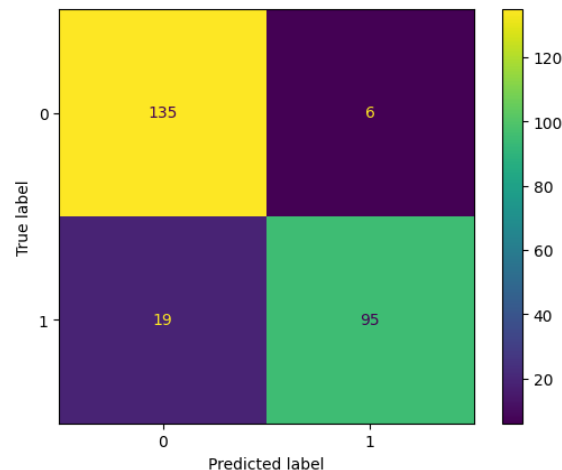
- The average of the results of the Neural Network and Gradient Boosting models (best performance)

```
accuracy = 0.9019607843137255  
precision = 0.9405940594059405  
recall = 0.8333333333333334
```

- We get a model with **similar performance as the other two** (maybe with a little better metrics, but the testing made is not very conclusive) → it can be a good option in terms of raw performance that we combine two models (or more) **BUT**:

the gain in performance doesn't justify the use of two different models to get results → instead of training and optimizing a model we need to make it two or more times, which means more resources and more time needed to get similar results.

- So, we can simply put a little **more effort into optimizing one of the models**, and in the end, **we get probably similar performance** utilizing the least resources and less complexity.
- ❖ Another thing is that **we can't guarantee that we get a performance boost** (precision and accuracy increased, but the recall decreased)



Result Summary

Model Data	KNN	Naive Bayess	Decision Tree	Random Forest	Gradient Boosting	SVM	Neural Network	Average model
Unbalanced Data	A - 0.90 P - 0.93 R - 0.32	A - 0.91 P - 0.76 R - 0.49	A - 0.94 P - 0.86 R - 0.74	A - 0.96 P - 0.95 R - 0.78	A - 0.96 P - 0.94 R - 0.79	A - 0.88 P - 0.92 R - 0.20	A - 0.97 P - 0.94 R - 0.83	-
Balanced Data	A - 0.81 P - 0.88 R - 0.73	A - 0.80 P - 0.79 R - 0.82	A - 0.87 P - 0.93 R - 0.80	A - 0.90 P - 0.94 R - 0.85	A - 0.88 P - 0.90 R - 0.86	A - 0.69 P - 0.84 R - 0.48	A - 0.88 P - 0.91 R - 0.84	A - 0.90 P - 0.94 R - 0.83

Legend:

A → Accuracy

P → Precision

R → Recall

Evaluation and Main Conclusions

1. What is the best model and the recommended data science procedure for the business?

This boils down to a **balance between performance, scalability and interpretability**:

Interpretability: Decision Trees;

Overall: Neural Networks

2. What do you think the business can gain from your data science effort?

Predicting with satisfactory degree of precision if a client will churn or not given the data that the business has in its hands about that specific client. With these potential predictions, **decisions can be made to retain the clients that the model predicts will churn and potentially have a monetary gain.**

Also, the results of the model can give a very useful insight into **what makes people churn** so it can also **make changes to how the business operates to decrease even further the churning rate.**

To wrap up the resulting model is not very computationally expensive to build and run so the **cost of constructing, maintaining and operating** the model would probably be **minimal** compared to the **potential gains.**

3. Lessons learned

- **Data Processing:** Regarding this, we applied and combined different procedures to get an appropriate "version" of the dataset ready for each model.
- **Balanced Dataset Considerations:** To construct the balanced dataset, we simply discarded random points from the majority class (not churn) to match the number of points of the minority class
- **Modelling and Validation:** After applying all the proposed models to our data, our first necessity was to improve them.



References

- Jorge, Alípio Mário Guedes & Ferreira, Pedro Gabriel Dias. (2023). *Introdução à Ciência de Dados*. [PowerPoints de apoio à disciplina, lecionada na FCUP, MDS].

Packages:

- Numpy - <https://numpy.org>
- Pandas - <https://pandas.pydata.org>
- Matplotlib - <https://matplotlib.org>
- Seaborn - <https://seaborn.pydata.org>
- Sklearn - <https://scikit-learn.org/stable/>
- Mixed-naive-bayes - <https://pypi.org/project/mixed-naive-bayes/>
- Python - <https://www.python.org>