REPORT

CHOOSING THE ALGORITHM

While choosing the algorithm, it was taken into consideration that decision trees are useful for customer segmentation and that the effect of features on the result can be analyzed visually. For these reasons, it was decided to train a decision tree.

CHOOSING THE SUCCESS METRIC

When choosing the success metric to be optimized, the class imbalance of the dataset and the desire to focus on customers who will buy the investment product were taken into account. Optimizing accuracy would be misleading considering the class imbalance. It would be logical to optimize recall as it was desired to have a high true positive rate, but this method was not used because only optimizing the true positive rate could decrease accuracy. Therefore, it was decided to optimize the roc-auc score, assuming that true negative rate and true positive rate are equally important.

METHODOLOGY

- The day feature was divided into 3 categories to keep the number of features low.
- In order to remove the negative effect of the class imbalance, the class weight parameter was calculated and given to the model.
- Although it was not desired to separate the test data and measure the performance of the model on the test data, 20% of the data was reserved for the test in order to analyze on the confusion matrix.
- After training the decision tree visualized to analyze it.

RESULTS

- After grid search cross validation, the model with the best roc-auc score was selected. Best roc-auc score was 93.2%.

```
Fitting 5 folds for each of 4420 candidates, totalling 22100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed:    8.2s
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed:   13.6s
[Parallel(n_jobs=-1)]: Done 1234 tasks      | elapsed:   21.7s
[Parallel(n_jobs=-1)]: Done 1784 tasks      | elapsed:   32.2s
[Parallel(n_jobs=-1)]: Done 2434 tasks      | elapsed:   46.0s
[Parallel(n_jobs=-1)]: Done 3184 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 4034 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 4984 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 6034 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 7184 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done 8434 tasks      | elapsed:  4.0min
[Parallel(n_jobs=-1)]: Done 9784 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done 11234 tasks      | elapsed:  5.8min
[Parallel(n_jobs=-1)]: Done 12784 tasks      | elapsed:  6.3min
[Parallel(n_jobs=-1)]: Done 14434 tasks      | elapsed:  7.0min
[Parallel(n_jobs=-1)]: Done 16184 tasks      | elapsed:  7.9min
[Parallel(n_jobs=-1)]: Done 18034 tasks      | elapsed:  9.0min
[Parallel(n_jobs=-1)]: Done 19984 tasks      | elapsed: 10.2min
[Parallel(n_jobs=-1)]: Done 22034 tasks      | elapsed: 11.4min
[Parallel(n_jobs=-1)]: Done 22100 out of 22100 | elapsed: 11.5min finished
roc auc:  0.9320504318612287
Best Params:  {'criterion': 'gini', 'max_depth': 12, 'min_samples_leaf': 20, 'min_samples_split': 350}
Out[30]: 1
```

- The selected model reached accuracy by evaluating 5-fold cross validation is: 85.05%. It is above the desired success.

```
In [37]: scores = cross_val_score(dt, x_train, y_train, cv=5, scoring='accuracy')
    ...: avg_acc=np.mean(scores)
    ...: print("Average cross validation accuracy: "+ str(avg_acc))
Average cross validation accuracy: 0.8504999776031488
```

Bonuses:

- The image (decision_tree.svg) in the repository can be analyzed to find the customer segments that are likely to buy the product. We can say that the decisions made when going to the leaf nodes where the green color is dominant, show the customer profile that is likely to buy the product. If the following path is followed from root to leaf, a desired customer segment can be reached: right->right->right->left->right->left. Features of this segment are: duration > 966 and marital=single and age < 55. Another customer segment can be reached by following the following path: right->right->right->right->left->right->right. Features of this segment are: duration > 827 and marital=married and contact is not unknown and balance > 2920. If the tree is analyzed, more customer segments can be found.

- When the feature importances of the trained model is listed, it can be seen which feature affects the result most.

```
In [59]: print(dict(sorted(zip(dt.feature_importances_,x.columns))))
{0.0: 'month_dec', 0.00012176737670332121: 'month_aug', 0.00020689656480595562: 'job_retired',
0.00030644099001093575: 'job_student', 0.0003198073004225949: 'loan_yes', 0.0003247742914673392:
'job_technician', 0.0006087345843266832: 'marital_married', 0.000732889753492605: 'day_10-20',
0.0010940096918418622: 'job_self-employed', 0.0010953378020943081: 'month_jul', 0.001211486869826189:
'job_admin', 0.0015155587326944114: 'month_nov', 0.0024530413063942606: 'contact_telephone',
0.002656762882507133: 'month_jan', 0.002965777713469829: 'contact_cellular', 0.004696736416036059:
'education_tertiary', 0.005420778597701507: 'age', 0.005862686414738101: 'balance', 0.007111570921639999:
'campaign', 0.01621034414089461: 'month_may', 0.02045847760714099: 'day_20-31', 0.02304268566059977:
'contact_unknown', 0.023068882653599007: 'month_jun', 0.02618432772523918: 'day_0-10', 0.029871372058016155:
'month_oct', 0.03109856167595769: 'housing_yes', 0.0443198002662034: 'month_feb', 0.06717079638223492:
'month_apr', 0.07005933225219166: 'month_mar', 0.6098103613677496: 'duration'}
```

'duration' feature is the feature that affects the result the most. It is also located at the root of the decision tree. So it will be useful to focus on the 'duration' feature.

Additional Results:

- Confusion Matrix of model's predictions for test dataset:

	0	1
0	6305	1122
1	53	520

- Scores of model's predictions for test dataset:

```
In [54]: print("Test roc auc score: "+str(roc_auc_score(y_test,dt.predict(x_test))))
    ...: print("Test accuracy score: "+str(accuracy_score(y_test,dt.predict(x_test))))
    ...: print("Test recall score: "+str(recall_score(y_test,dt.predict(x_test))))
Test roc auc score: 0.8782169721296593
Test accuracy score: 0.853125
Test recall score: 0.9075043630017452
```