Predicting employee churn

Machine learning engineer nanodegree
Capstone Project proposal

Dante Ruiz

1. Project definition

1.1 Overview

- A company that relies on Big Data wants to hire data scientists among candidates that sign up for a special training they provide.
- The objective is to predict the probability that an enrollee will want to work for the company after the training or will start looking for a new job.
- This information is useful for the company as it helps to plan training courses as to reduce the cost and time and control for the quality of sessions.
- Current credentials, demographics and experience data is provided for each enrollee, as well as a target variable that indicates whether a candidate is not looking for a new job (0) or is looking for a job (1).
- The dataset comes from Kag gle.

1.2 Problem

- Estimate the probability that a candidate will start looking for a new job after the training.

1.3 Solution

- A machine learning XGboost model will be fitted on the data to estimate the probabilities of enrollees looking for a new job after the completion of their special training.

1.4 Metrics

- The model evaluation metric will be the Area Under the Receiver Operating Characteristic Curve (ROC AUC)

- This metric was chosen for two reasons:

    - First, because it was specified in the Kaggle task as the employee churn dataset provided is highly class imbalanced. There are more employees that are staying in a company than those leaving.

  - During model evaluation, AUC scores will be benchmakred against other models trained on the same data.

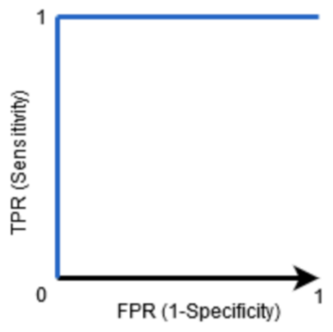
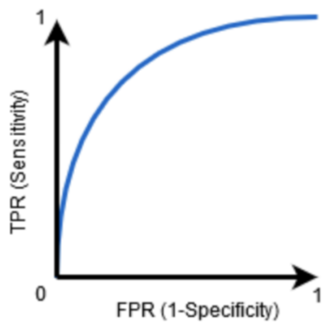
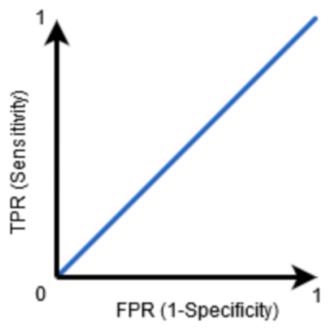
- Second, because it is a metric that deals well with target imbalance as it is insensitive to changes in class distribution.

  - The ROC AUC score is the area under the ROC Curve. Such curve is an evaluation metric of binary classification problems that summarizes the sensitivity and specificity of a classification model at different probability cut off points.

  - Suppose a binary classification problem with two classes Positive and Negative. The following table shows different theoretical ROC curves and their respective AUC score.

  - The higher the ROC AUC score, the better performance of the model at distinguishing between the Positive and Negative classes.

Example of ROC and AUC scores

AUC = 1	*AUC between 0.5 and 1*	*AUC = 0.5*
		

Source: Bhandari, A. 2020

  - When the AUC = 1, the classifier is able to perfectly distinguish between all Positive and Negative class points correctly. If the AUC = 0 then the classifier predicts all Positive as Negative and all Negative as Positive.
  - When the AUC between 0.5 and 1, there is a high chance that the classifier will be able to distinguish the Positive class from the Negative class, because the model is able to detect more numbers of True Positives and True Negatives than False Negatives and False Positives.
  - When AUC = 0.5, the classifier is not able to distinguish between Positive and Negative class points, meaning either the classifier is predicting classes in a random fashion or using a constant value.

2. Exploratory data analysis

In this section, the results of the exploratory data analysis is presented.

2.1 Features

The dataset contains the following 13 features about demographic and professional experience characteristics of each enrollee. It also includes a target variable that describes whether a candidate is looking for a new job or not.

- enrollee_id: Unique ID for candidate
- city: City code
- city_development_index: Developement index of the city (scaled)
- gender: Gender of candidate
- relevent_experience: Relevant experience of candidate
- enrolled_university: Type of University course enrolled if any
- education_level: Education level of candidate
- major_discipline: Education major discipline of candidate
- experience: Candidate total experience in years
- company_size: No of employees in current employer's company
- company_type: Type of current employer
- lastnewjob: Difference in years between previous job and current job
- training_hours: training hours completed
- target: 0 – Not looking for job change, 1 – Looking for a job change

2.2 Shape of data sets

- The shape of rows and columns of the training and testing data is presented below:

Shape of datasets

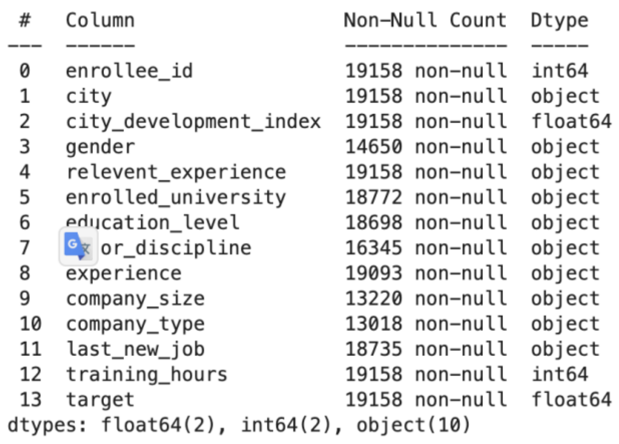
	Rows	Columns
Train	19158	14
Test	2129	13

- The testing set does not include the target variable for prediction. However, it was possible to get the true targets from a different source, which will be used for model benchmarking.

2.3 Fields and data types

- 10 out of 14 fields are categorical and the reamining 4 are numerical.

Fields by data type

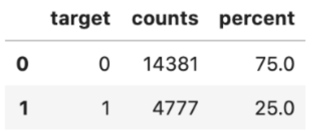


```
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19158 non-null  int64  
 1   city                    19158 non-null  object 
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object 
 4   relevent_experience     19158 non-null  object 
 5   enrolled_university     18772 non-null  object 
 6   education_level         18698 non-null  object 
 7      or_discipline        16345 non-null  object 
 8   experience              19093 non-null  object 
 9   company_size            13220 non-null  object 
 10  company_type            13018 non-null  object 
 11  last_new_job            18735 non-null  object 
 12  training_hours          19158 non-null  int64  
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
```

2.4 Target descriptions

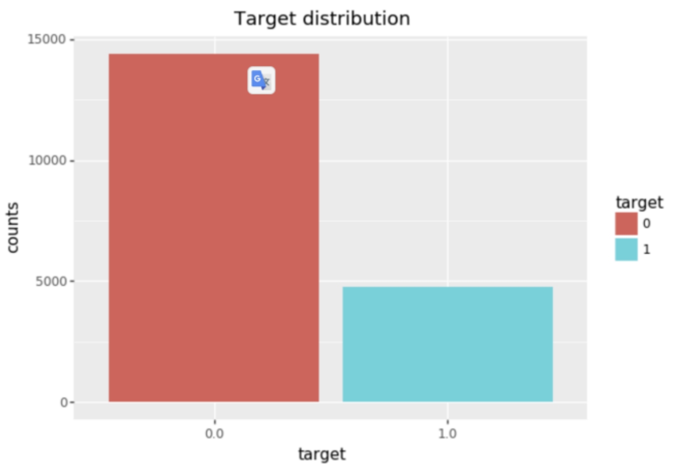
- The problem of employee turnover is characterized for having an unbalanced target variable. Here it is not the exception, 25% of the enrollees decided to look for a new job outside the company.

Target variable distribution



	target	counts	percent
0	0	14381	75.0
1	1	4777	25.0

- It is a highly imbalanced target, for each enrollee that wants to look for a new job, there are three that are staying.

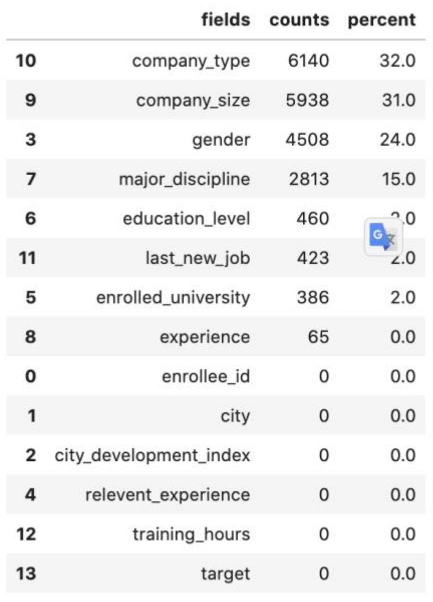


- The training and validation datasets will have to be splitted using the target variable for stratification to correct for imbalance.

2.5 Missing values

- There are missing values in the dataset that have to be investigated.

Missing values by feature



	fields	counts	percent
10	company_type	6140	32.0
9	company_size	5938	31.0
3	gender	4508	24.0
7	major_discipline	2813	15.0
6	education_level	460	2.0
11	last_new_job	423	2.0
5	enrolled_university	386	2.0
8	experience	65	0.0
0	enrollee_id	0	0.0
1	city	0	0.0
2	city_development_index	0	0.0
4	relevent_experience	0	0.0
12	training_hours	0	0.0
13	target	0	0.0

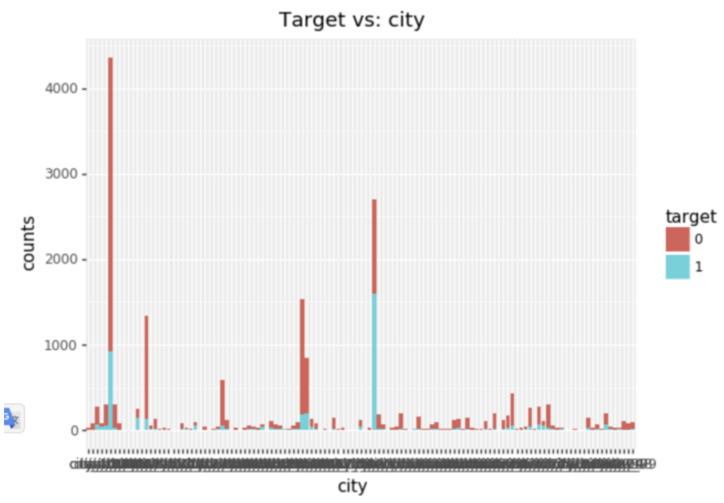
- The fields company_type, company_size, gender and major_discipline are the fields that have more than 15% of their values missing.

- Education_level, enrolled_university and last_new_job have only 2% values missing.

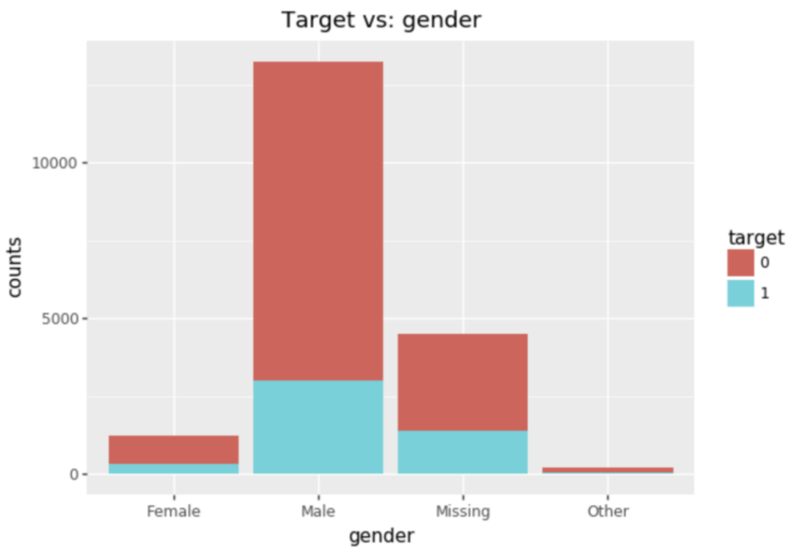
- An imputation strategy should be used such as the mode or trying aggregate them in other category.

2.6 Visualizations

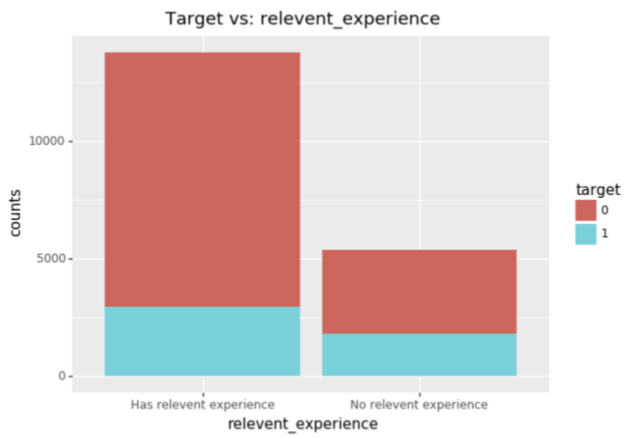
- In this section, each of the variables in the dataset is investigated using data visualizations.
    - **city**: It is very hetrogeneous, there are 6 cities with more observations than the others.



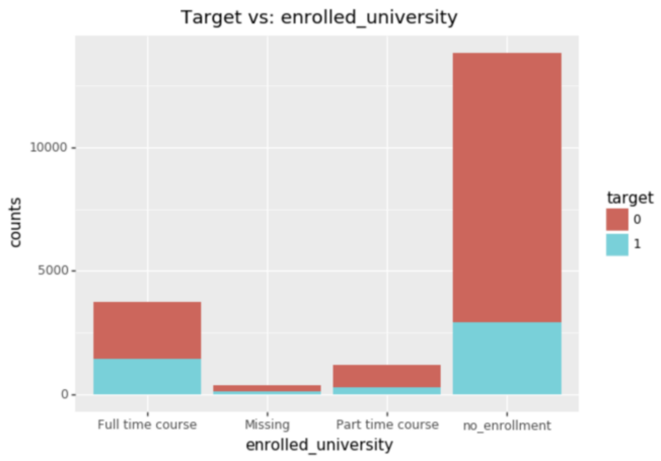
    - **gender:** There are more males than females, but also missing genders is high enough. Missing genders could be aggregated with others, but in this context they will be kept as separate categories.



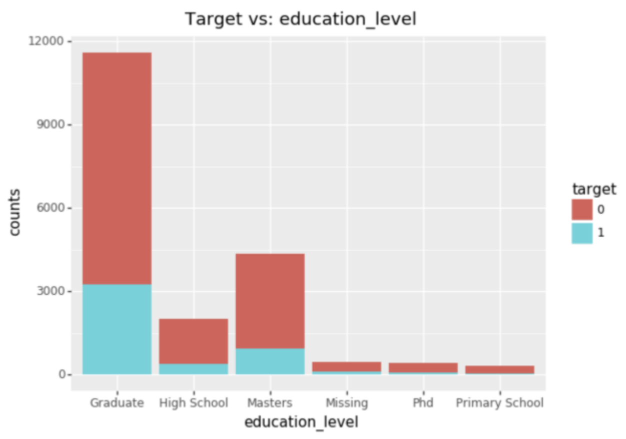
- **relevant_experience:** There is people with more relevant experience than others, but in terms of who wants to stay and who wants to leave there is no clear pattern.



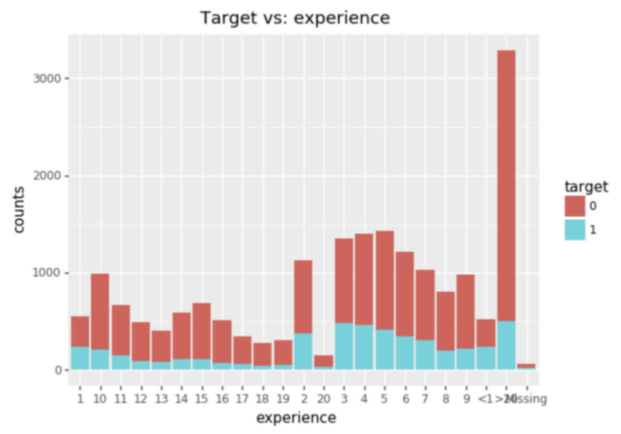
- **enrolled_university**: There are more enrollees not enrolled in university, I will assume they are professionals. Missing values can be aggregated to no_enrollment.



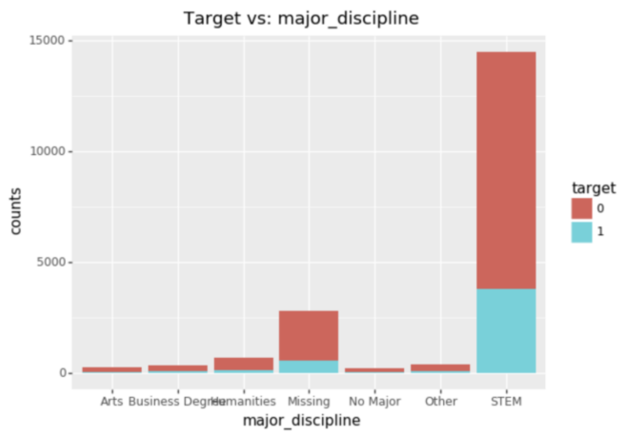
- **eductaion_level**: Most of the enrollees are graduates and have a postgraduate. There are many other education levels that could be aggregated to have more general categories.



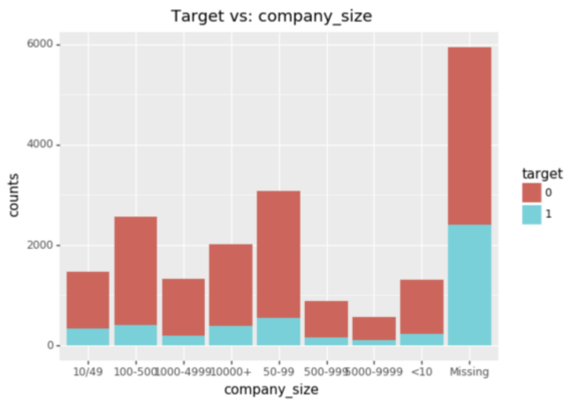
- **experience**: There are more enrollees with less than or equal to 9 years of experience than with more than 9 years of experience. It is more likely that people with less years of experience choose to look for a new job. This variable could also be aggregated into more general values.



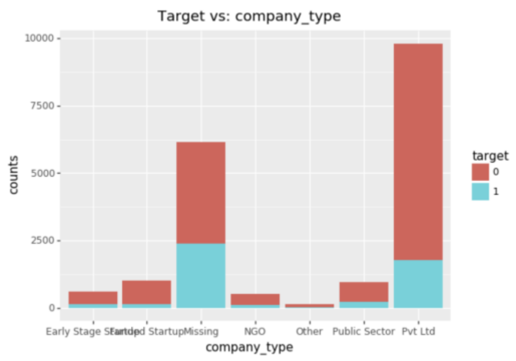
- **major_discipline**: Most enrollees come from STEM major disciplines. The rest show very few records. Major disciplines could be aggregated into more general ones.



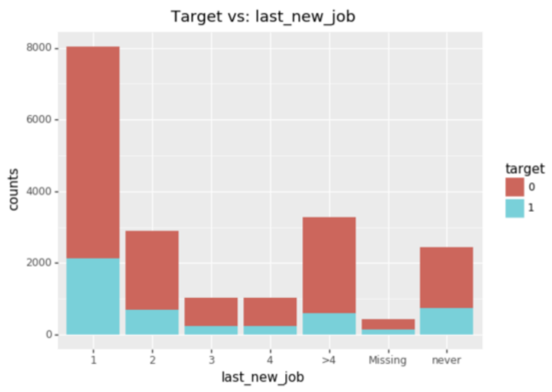
- **company_size**: Most other enrollees did not indicate company size, they could be aggregated into the less than 10 category. This field is a bit dirty; it could be aggregated into more general categories. Apparently, enrollees with missing values tend to look for a new job.



- **company_type**: Most of the enrollees work for private sector, however the number of records with missing data are too many. Missing and other company type could be aggregated. The rest of the types could be aggregated into more general ones.



- **last_new_job**: Most of the enrollees are on their first job. It less likely that people with more years of experience look for a new job. Missing and never could be aggregated.



3.Preprocessing

3.1 Clean data

- Aggregate categories within fields.
- Missing values are aggregated as descripted above.
- Converts all values into lower case and removes spaces.
- The following aggregation decisions were made for each feature:

Aggregating descions by field

Fields	Descisons
city	-
city_development_index	-
gender	-
relevent_experience	-
enrolled_university	NA are relabeled as missing values
education_level	• NA, primary_school, high_school are relabeled as not graduate. • Masters, PHD are relabeled as post_graduate
major_discipline	• Humanities, business_degree, arts are relabeled as not_stem • No major and NA are relabeled as others.
Experience	The years of experience are aggregated as follows: • less_or_equal_than_5

	• less_or_equal_than_10 • less_or_equal_than_15 • less_or_equal_than_20 • greater_than_20
company_size	The company size is aggregated as follows: • less_or_equal_than_99 employees • less_or_equal_than_999 employees • greater_than_999 employees • missing
company_type	• funded_startup, early_stage_startup are relabeled as startup • public secto, ngo are relabeled as government_or_ngo • no_major and NA are relabeled as other
lastnewjob	Last new job is aggregated as: • Never and <1 are relabeled as less_or_equal_than_1 • less_or_equal_than_2 • less_or_equal_than_4 • more_than_4

3.2 Select features to be included in the design matrix

- All variables are selected except enrollee id and city. For city we are using the city_development_index as it is more informative.
    - Categorical fields: gender, relevent_experience, enrolled_university, education_level, major_discipline, experience, company_size, company_type, last_new_job
    - Numerical fiels: city_development_index, training_hours

3.3 One Hot Encode categorical variables.

- As much of the variables are categorical, each variable is one hot encoded.
- After one hot encoding there are 33 fields.

3.4 Split the training data into training and validation data

- Training is 75%
- Validation is 25%
- Splitting is stratified by the target variable to preserve the same proportions in training and validation dataset.

4. Algorithms and Techniques

- In this section the model and techniques applied to the problem are briefly explained

4.1 Model choice:

- The model choice for this problem is the XGboost algorithm that stands for "Extreme Gradient Boosting".
- For supervised classification problems such as employee churn, XGboost is state of the art. Below there is a list of strengths and weaknesses.

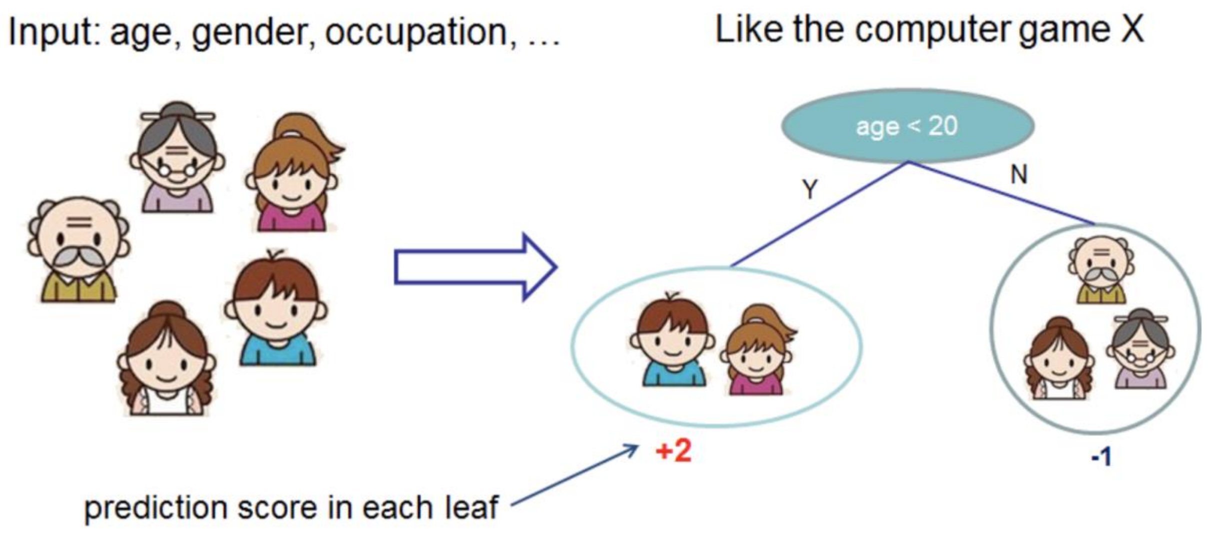
XGboost strengths and weaknesses

Strengths	Weaknesses
▪ Consistently outperforms other models ▪ It has won recently most Kaggle competitions ▪ Speed and performance. ▪ Parallelizable algorithm in many computers ▪ Works best on structured data	▪ Without regularization they tend to overfit very quickly. ▪ Complex non-linear models ▪ Difficult for interpretability. ▪ On large datasets it takes time to train, but less than the vast majority of conventional algorithms.

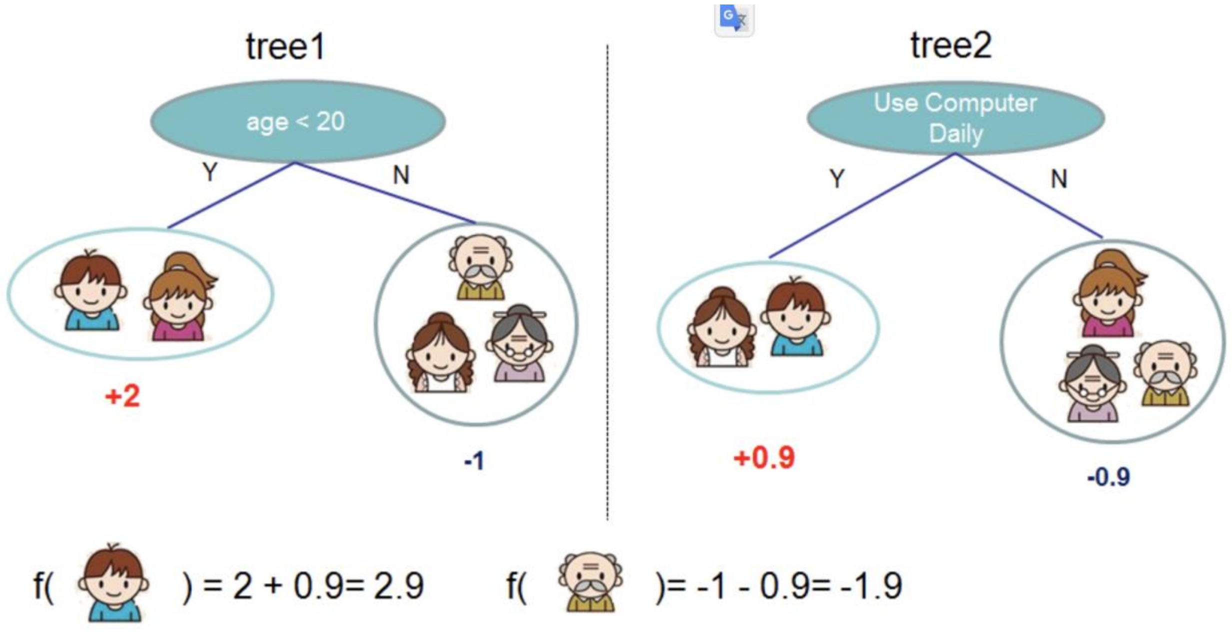
4.2 Model overview

- It uses trees as decision base learners, more specifically a classification and regression trees (CAR)T. A CART is composed of a series of binary questions (YES/NO) that eventually yield a prediction at the leaves of the tree. CARTs use scores for predictions rather than classes as simple decision trees.
- Below there is an example of a CART that classifies whether someone in a family will like a video game[1].

[1] Introduction to Boosted Trees. https://xgboost.readthedocs.io/en/latest/tutorials/model.html


Input: age, gender, occupation, …
Like the computer game X
age < 20
Y
N
+2
-1
prediction score in each leaf

- XGBoost is an ensamble model which means that usually, a single tree is not strong enough to be used in practice. What is actually used is the ensemble model, which sums the prediction of multiple trees together. An example of two trees is provided below:



- For each member of the family the score it gets in each tree is summed to get the final score. Each tree complements each other.
- Mathematically speaking we can rewrite the model as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

    - K = number of trees
    - f is a function of functional space F.
    - F = is the set of all possible CARTs
- The objective function to be optimized is:
    - Theta = parameters
    - The l function is the loss function .
    - The omega function is the regularization term.
    - Y_hat stands for the prediction value of the i-th element at time t.

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

- The xgboost used boosting to train the model. Boosting is a meta-algorithm that converts many weak learners (Machine learning algorithms that are slightly better than chance) into a strong learner. A strong learner is any algorithm that can be tuned to achieve good performance.
- To train the model an additive strategy: fix what has been learned and add a new tree one at a time. This iteratively strategy is learning from a set of weak models on subsets of data, weighing each weak prediction according to each weak learner's performance (it is not linear). This is much better than the individual predictions themselves.
- Below is a diagram that explains how boosting works in this context. On the left the residuals and on the right the predictions vs the observed values in the training set. Each row is a new model that is built upon the residuals of the previous model. The idea is that after adding new trees to the model, any function can be approximated non linearly.

Gradient Boosting

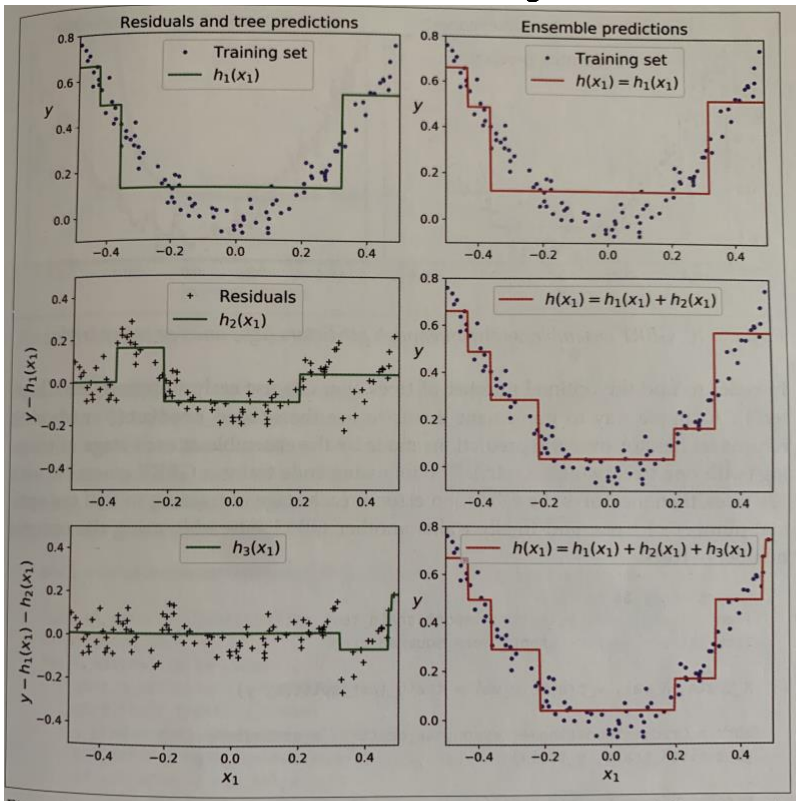


Figure 7-9. In this depiction of Gradient Boosting, the first predictor (top left) is trained normally, then each consecutive predictor (middle left and lower left) is trained on the previous predictor's residuals; the right column shows the resulting ensemble's predictions

Source: Aurélien Géron, 2020

- Finally, predictions are non-linear compared to a linear logistic model.

4.3 Hyperparameters

- The xgboost uses the following hyperparameters to tune and regularize the model.
  - Eta: Learning rate controls how quickly the model fits the objective function using additional base learners (between 0 and 1).
  - Max_depth: How deeply each tree is allowed to grow during a boosting round (positive number).
  - Subsample: The fraction of your training data used per boosting round (between 0 and 1). If it is very small the model runs into underfitting problems, while using all runs into overfitting problems.
  - Colsample_bytree: The faction of features used per tree. A large value means that almost all features can be used to build a tree during a given boosting round
  - Gamma: Minimum loss reduction allowed for a split to occur
  - Alpha: L1 regularization of the weights, larger values mean more regularization
  - Lambda: L2 regularization on leaf weights.

4.4 Hyperparameter tunning.

- It is used to find the optimal configuration of hyperparameters that lead to the lowest loss possible.
- A bayesian optimization strategy is used to look for the best parameters. This technique builds a probability model of the objective function and uses it to select the most promising hyperparameters to evaluate in the true objective function.
- This technique is used instead of exploring all possible combinations using a grid search strategy, or exploringly randomly some parameters in the grid search.
- A Bayesian strategy allows hyperparameter tuning to exploit the best-known results. It chooses the best hyperparameters for the next training job by considering everything that it knows about the problem so far.

4.5 Cross validation

- XGboost uses cross validation to estimate the performance of a model on unseen data. It generates many non-overlappping train/test splits on training data. It reports the average across all data splits.

4.6 Tuning evaluation metric:

- Two evaluation metrics are used to tune the model fit accuracy and ROC AUC.
- This means that for each configuration the model that is picked is the one that maximizes the accuracy or the ROC AUC score.
- Then results are compared against the ROC AUC score.

5. Modelling

5.1 Amazon Sagemaker

- For the implementation of this project Amazon Sagemaker platform and their implementation of the Xgboost is used.
- Cleaned training, validation and testing data is uploaded to S3.
- An ml.m4.xlarge instance machine is used to train and test the model
- An xgboost v1.2-1 amazon container is used.

5.2 Hyper parameter tuning

- To control overfitting, hyper-paramater tuning is applied choosing from a reasonable range of values for max depth, eta, min child weight, gamma and subsample. This hyperparameters were chosen to yield a reasonable regularized model.

Hyperparameters and range of values

Parameter	Range of values
max_depth	From 3 to 8
eta	[0.001, 1]
gamma	[0, 5]
min_child_weight	From 2 to 6
subsample	[0.5,0.9]
colsample_bytree	[0.5,1]

- 20 different combinations were tried using the Hyper Parameter Grid Seach using a Bayesian strategy.
- The best estimator is used to create a batch transform job to test our model. Deployment of the model in a web app is not considered, as the business will provide batch data every period of time, on a low frequency period.

5.3 Model training

- Two xgboosts were fitted each tuned by a different objective metric: accuracy and ROC AUC score.
- The best models for each tuned model are dispayed in the table below.

Best Models Hyperparameters

Hyperparameter	Range type	Tuned by accuracy	Tuned by ROC AUC score
tuning objective metric	FreeText	validation:accuracy	validation:auc
colsample_bytree	Continuous	0.644046136	0.769398377
eta	Continuous	0.001413778	0.011194137
gamma	Continuous	3.055048956	3.497174126

max_depth	Integer	4	3
min_child_weight	Integer	3	6
num_round	FreeText	1000	1000
objective	FreeText	binary:logistic	binary:logistic
subsample	Continuous	0.530590121	0.621283305

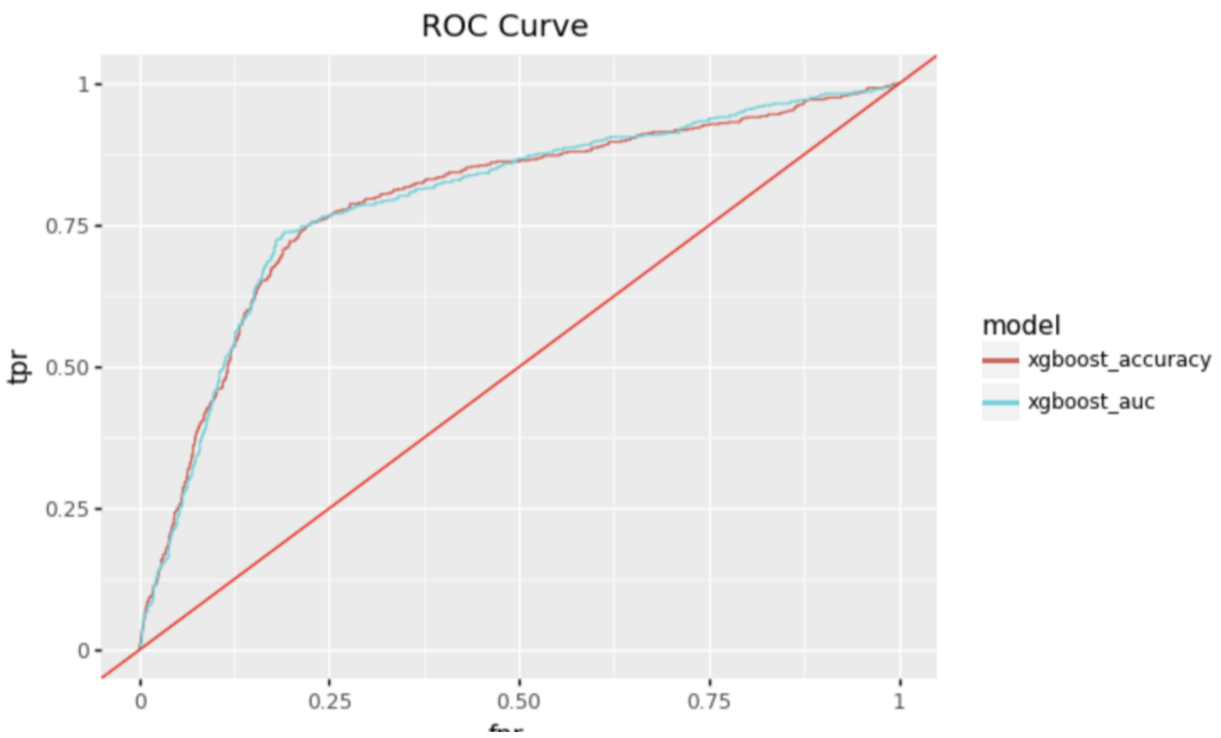
6. Evaluation

- In this section we compare the results of both xgboost probability predictions for enrollees looking for a job change. The established comparison criteria is the Area Under the Curve (AUC) as stated in the Kaggle's problem description.

- The AUC of these models is compared to the AUC scores obtained by Joshua Swords using the following models: SVC, Decision Tree, Random Forest, Logistic Regression and KNN.

6.1 ROC Curves and AUC

- Estimating the ROC curves, it can be seen that both models a very similar classifier. Their AUC curves are almost perfectly overlapped.
- At this point it can be said that both models will do the job of yielding very similar probability predictions for employee turnover.



- Below the AUC score is calculated, and it can be seen that both models yield 79% of AUC score, where 100% is the maximum.

ROC AUC scores

Model	ROC AUC score
xgboost tuned for better accuracy	0.79
xgboost tuned for better AUC score	0.79

6.2 AUC scores vs Benchmarks

- The following table contains the model performance metrics that Joshua Swords obtained using SVC, Decision Tree, Random Forest, Logistic Regression and KNN using the same data. precision and ROC AUC score. In each row he presents the scores of each model conditional on the evaluation metric.

Model benchmarking by AUC score

Models	ROC AUC score
SVC	**65%**
Descision Tree	**63.3%**
Random Forest	**68.4%**
Tuned Random Forest	**54.3%**
Logistic Regression	**63%**
KNN	**63.9%**
XGoosts Tuned for Accuracy and AUC score	**79%**

- The ROC AUC scores he obtained are in the range of 54.3% to 68.5%. The best performing model was a Random Forest (in yellow).

- The XGboost model implemented in this project is well known to achieve superior results than any of the rest of the models presented in the table above.

- The ROC AUC scores of the two XGboosts estimated (in green) outperformed by almost 10% points the Random Forest.

- The XGboosts were tuned for maximum accuracy and ROC AUC score after hyper parameter tuning. Both models seem to be almost identical as how they separate both classes, no matter the threshold used.

- Both models can be implemented for the task at hand.

- At this point, whether these models are good enough, will depend on how useful they will be for Human Resources making decisions in the real world.

7. Conclusions

- In this project, machine learning is used to help a given company that is specialized in Big Data, to estimate the probabilities of enrollees with special training that will look for a new job outside the company after completing their courses.

- Two XGboost models are fitted and tuned to obtain the best predictions for probabilities. Compared to other models trained on the same dataset, the Xgboost models outperformed the other by almost 10% points.

- Both XGboost models yield the same AUC score, so any of them could be used.

- In the process of estimation sagemaker built in models were used, and hyperparameter grid search was applied to control for overfitting.

- The training set was split into a validation set, in order to ensure that appropriate model validation was applied.

Depending on the costs that the company incurs on recruiting candidates and imparting training sessions, they can use these probabilities for decision making.

- Segment their enrollees and work with the best candidates that wish to look for a new job, in order to retain them.
- Improve their training in order to reduce the probabilities of looking for a new job outside the company.
- Focus further efforts on candidates that want to stay in the company.

These are some of the possibilities on how these probabilities can be used.