Machine Learning Engineer

Udacity Capstone Project

Bruno Marques

October 25th 2018

Definition

Project Overview

Currently a lot of companies suffers with employee's attrition, these turnovers causes business to loose qualified personal and valuable assets. Other problem that attrition rates causes are the knowledge lost, sometimes this can cost years of research or thousands of dollars in development to companies, so it is clearly a problem most business would like to avoid or to reduce.

When considering downtime, recruiting, interviewing, training, and getting to speed expenditures are substantial, according to Porter (2011). An entry-level position can cost between 50 to 100 percent of the employee's wage to the organisation (Porter, 2011).

In this project a set of employee data is analysed and used to train a prediction model that tells if an employee is likely to leave job in the current year.

Problem Statement

The proposed capstone project will assess and examine employee attrition rates of company A (real name was hidden due to confidentiality issues), studying employee records will supply supportive information. Obtaining insights from the data will clarify reasons that make employees to leave and help to provide some answers to the questions the company has, such as:

1. What could be changed in the work environment to change the mind of those who wish to leave?
2. What are the main reasons?
3. Is there a location with an accentuated turnover problem?

Presented data will help top management improve decision-making relating to employee work policies in addition to generating new insights on employees retention. The expected result is a diminishing employee turnover and a greater

talent retention, as well as lower expenditures with new employees qualification and training.

Metrics

As we are working with a classification problem, thus the response variable being binary a good metric is the accuracy. Accuracy measures the overall correctness of the model, it sums the total number of true trues and true falses and divide by the total amount of observations. As the employee attrition situation is not something critical it does note need to focus on recall (rate of true trues over the sum of true trues and true falses), neither need to focus on precision (rate of true trues over the sum of true trues and false trues).

Analysis

Data Exploration & Exploratory Visualization

The dataset holds information of 9 years of employees records, showing cases that are active and those that were terminated.

The datasets contain: employee id; employee record date (year of data); birth date; hire date; termination date; age; length of service; city; department; job title; store number; gender; termination reason; termination type; status year; status; business unit

The dataset has 6284 entries, employee's id, with over 18 features. A total of 1485 employees decided to leave the company, representing a turnover of 23%, of this percentage 915 (61%) were women and 570 (39%) men.

The plots below present the data analysed by different aspects. Those visualizations are helpful to identify patterns and tendencies in employee attrition.

Fig. 1 Age x count (frequency), separated by gender.

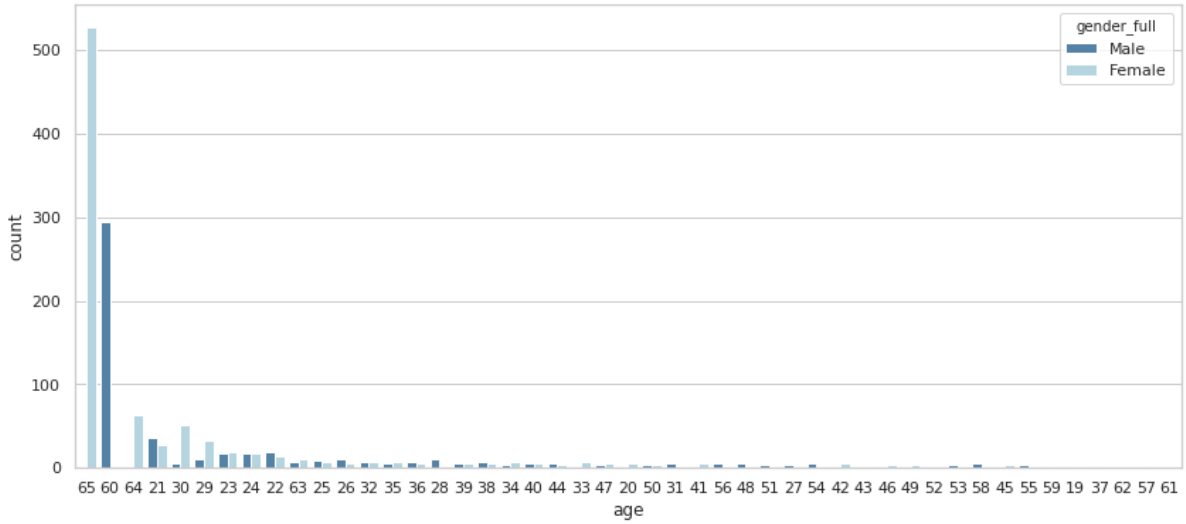


Fig. 2 Termination reason, separated by gender.

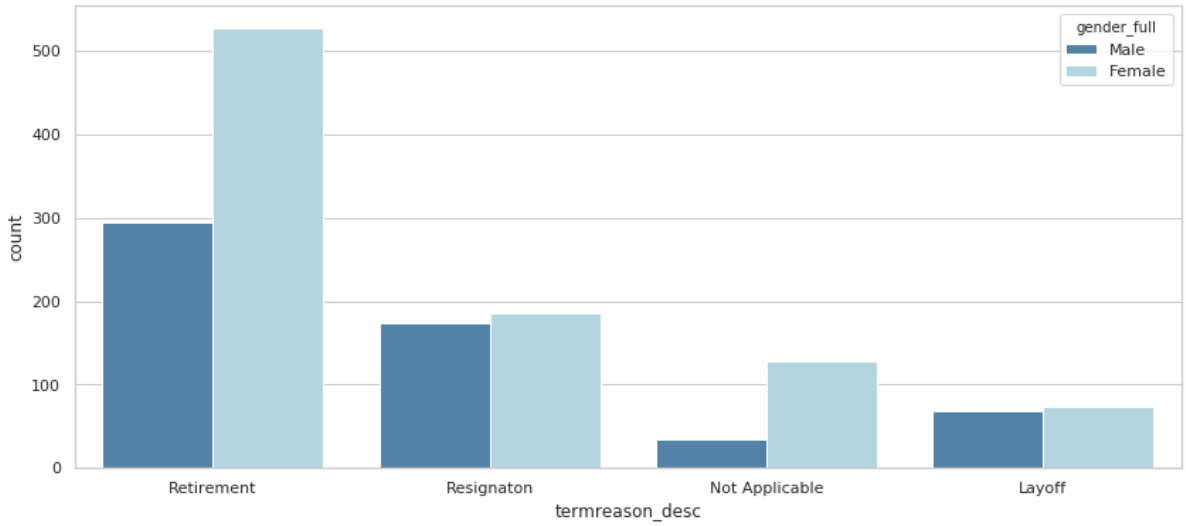
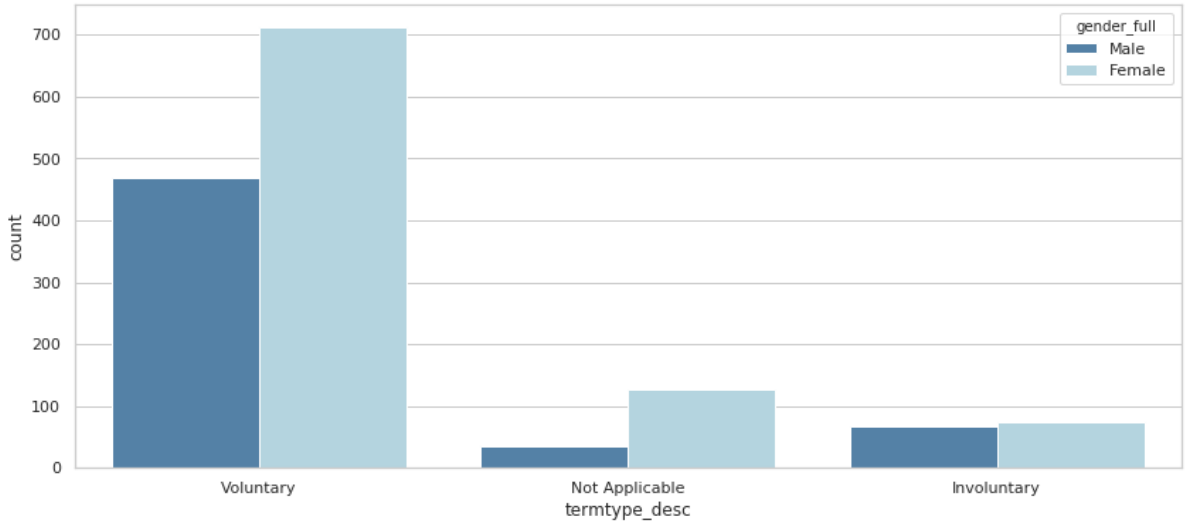


Fig. 3 Termination type, separated by gender.



Insights about the plots:

- Most women **voluntarily** leave for **retirement** at age of **65**;
- Most men **voluntarily** leave for **retirement** or by **resignation** at age of **60+**;
- Women represents the higher percentage of employee's turnover.

Below is a plot crossing the numerical features and following a heatmap, the objective is to identify correlations between features.

Fig. 4 Bi-variable analysis of features.

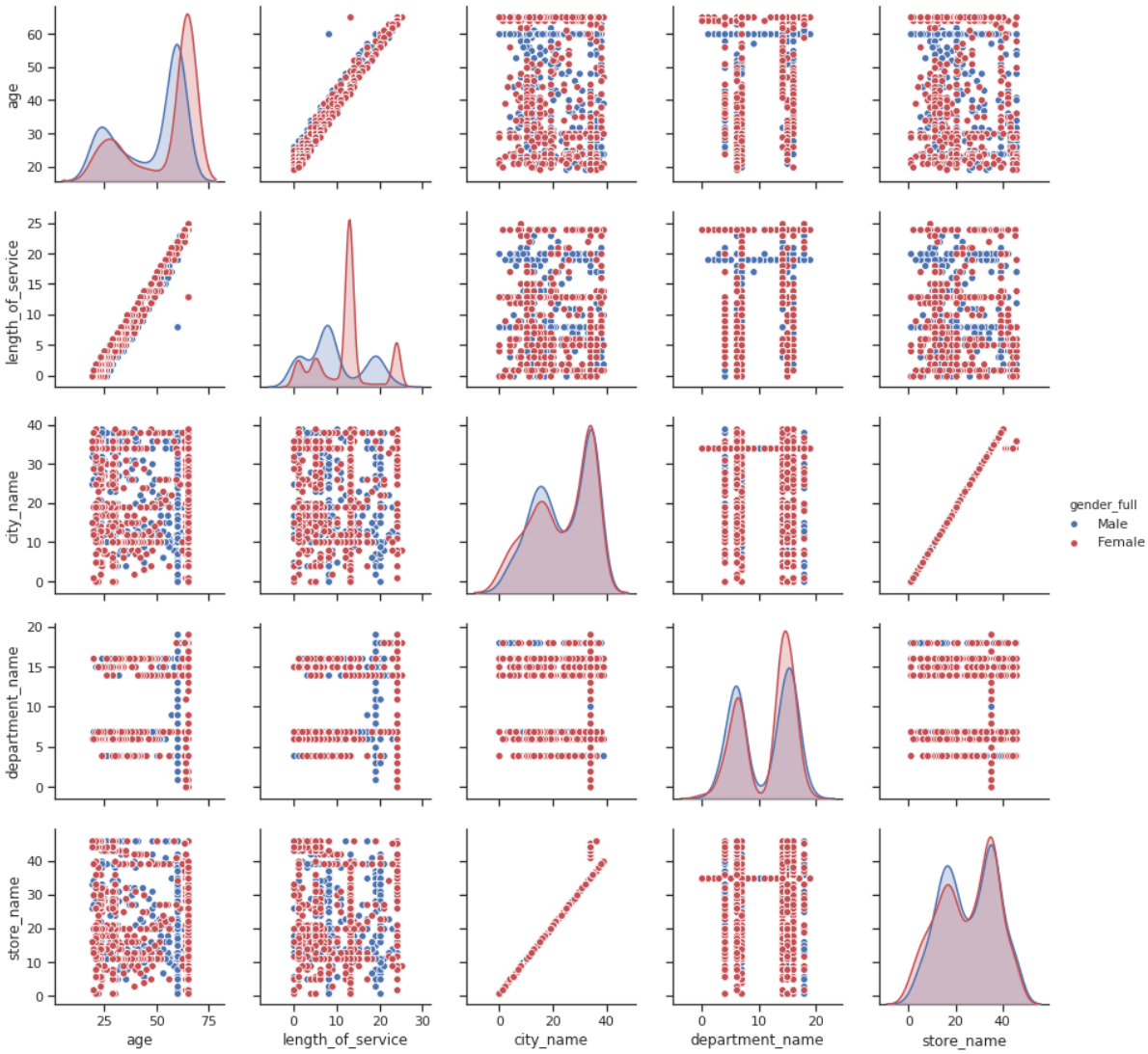
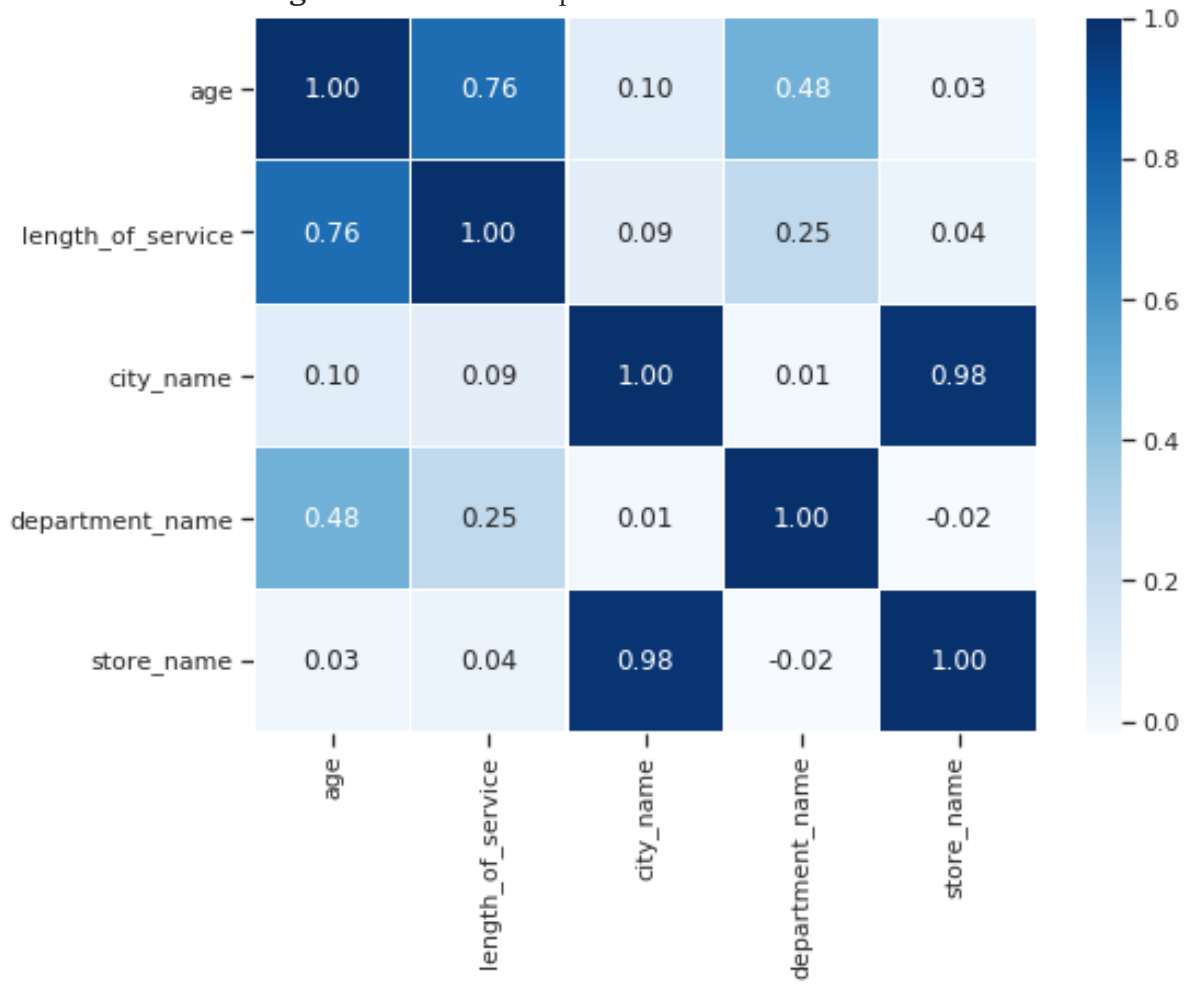


Fig. 5 Features heatmap.



Insights about the correlations:

- There is a positive linear relation between age and length_of_service, this means that for a model one of the two could be discarded as the other would carry the same information, the same applies to city and store;
- The heatmap shows the strength of the age & length_of_service, city & store pairs correlation, both strong (values above 0.7).

After assessing the dataset, using classical statistical tools, it is safe to assume a prediction model can be fitted into the data, it will be able to extract information and be properly trained in order to predict, with a good percentage, possible cases of employees attrition.

Algorithms and Techniques

Based on the data structure and feature's types, and as stated by Kashyap (2018), the Logistic Regression model is the best model for this type of problem. Logistic Regression is a simple model based on a linear equation, it separates the outputs with a single line, each side of the line represent a category. Besides it's name, Logistic Regression, is model often used in classification problems.

The following parameters can be tuned in a Logistic Regression model:

- penalty: used to define the type of penalisation, L1 or L2;
- C: regularization weight term;
- solver: algorithm to use in the optimization problem;
- max_iter: Maximum number of iterations taken for the solvers to converge.

Benchmark

Logistic Regression is a model based on linear equations that divides the outputs into two groups, this is a really good model for binary classification problems. As mentioned by Kashyap (2018) Logistic Regression presents the best accuracy for this type of dataset when compared to Lasso, Ridge, Decision Trees and Random Forests methods.

Methodology

Data Preprocessing

The data preprocessing strategy adopted for this dataset consists of the following techniques:

- identify main features;

- remove duplicated or of no use features;
- encode categorical features into numerical ones;
- group the data by employee id and use only the last year entry;
- segregate the target feature from the original dataset;
- balance the entries in the target feature by up sizing it with dummy entries;
- divide the data into train and test sets, 75% and 25% respectively.

The same techniques are applied to the prediction dataset, except for the last to methods that are specific for the learning phase.

Implementation

After finishing the Exploratory Data Analysis the model implementation process consisted of three main stages:

i. Train the selected model (Logistic Regression) using the train set;
ii. Tune (grid search) the model by testing different parameters;
iii. Evaluate metrics and define best parameters set-up.

At the train stage the standard model was presented to the train set, learning from it, next the model was evaluated regarding its accuracy, as shown in the image below:

img. 1 Model training

```
# test the different models and assess which one has the best score
pipeline = []
pipeline.append(('LR', Pipeline([('LR', LogisticRegression(solver='lbfgs'))])))

results = []
names = []
for name, model in pipeline:
    kfold = KFold(n_splits=10, random_state=7)
    cv_results_train = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results_train)
    names.append(name)
    msg1 = "Train result of model %s: mean accuracy - %f (std: %f)" % \
    (name, cv_results_train.mean(), cv_results_train.std())
    print(msg1)
```

```
Train result of model LR: mean accuracy - 0.848430 (std: 0.010633)
```

With the model properly trained and an initial accuracy score recorded it was submitted to a fine tuning process, using a method called “GridSearch”, the objective of this method is to identify the best parameters of the model (img. 2).

img. 2 Model fine tuning (GridSearch)

```
# perform a gridsearch to find the best parameters
params = {'penalty':['l2'],\
          'C': [0.01, 0.03, 0.05],
          'solver': ['lbfgs', 'liblinear', 'saga'],\
          'max_iter':[3000, 4000]}

model = LogisticRegression()
kfold = KFold(n_splits=10, random_state=7)
grid = GridSearchCV(estimator=model, param_grid=params, scoring='accuracy', cv=kfold, verbose=1)

grid_result = grid.fit(X_train, y_train)

means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:  2.9min finished
0.851625 (0.010949) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.01}
0.732981 (0.023753) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.01}
0.849819 (0.012247) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.01}
0.851625 (0.010949) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.01}
0.732981 (0.023753) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.01}
0.849819 (0.012247) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.01}
0.851625 (0.012096) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.03}
0.778550 (0.017176) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.03}
0.853293 (0.011391) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.03}
0.851625 (0.012096) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.03}
0.778550 (0.017176) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.03}
0.853293 (0.011391) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.03}
0.850792 (0.011619) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.05}
0.798833 (0.018640) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.05}
0.852598 (0.011530) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.05}
0.850792 (0.011619) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.05}
0.798833 (0.018640) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.05}
0.852459 (0.011485) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.05}
Best: 0.853293 using {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.03}
```

Finally the best set-up is evaluated, comparing results from the standard model and the fine tuned one, then the final model is defined, built and evaluated against the test set, as shown on image 3.

img. 3 Final model

```
model = LogisticRegression(penalty='l2', C=0.03, max_iter= 3000, solver= 'saga')
model.fit(X_train, y_train)

predictions = model.predict(X_test)
print (accuracy_score(y_test, predictions))
```

```
0.86375
```

Refinement

Indeed using the Logistic Regression presented a good result, as shown in the previous steps the final model achieved an accuracy score of over 86% in the test set. In fact the model presented a good result even with the standard

parameters, 84% during training step, converging with the analysis made by Kashyap (2018).

In order to improve the final accuracy score a grid search technique was used, this technique is widely utilized to fine tune models. The concept is quite simple, a dictionary with the different parameters and values ranges is created, next the model is tested with each distinct combination of parameters, at the same time it is being scored (img. 4).

img. 4 Grid search combinations and scores

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:  2.9min finished
0.851625 (0.010949) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.01}
0.732981 (0.023753) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.01}
0.849819 (0.012247) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.01}
0.851625 (0.010949) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.01}
0.732981 (0.023753) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.01}
0.849819 (0.012247) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.01}
0.851625 (0.012096) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.03}
0.778550 (0.017176) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.03}
0.853293 (0.011391) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.03}
0.851625 (0.012096) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.03}
0.778550 (0.017176) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.03}
0.853293 (0.011391) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.03}
0.850792 (0.011619) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 3000, 'C': 0.05}
0.798833 (0.018640) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 3000, 'C': 0.05}
0.852598 (0.011530) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.05}
0.850792 (0.011619) with: {'penalty': 'l2', 'solver': 'lbfgs', 'max_iter': 4000, 'C': 0.05}
0.798833 (0.018640) with: {'penalty': 'l2', 'solver': 'liblinear', 'max_iter': 4000, 'C': 0.05}
0.852459 (0.011485) with: {'penalty': 'l2', 'solver': 'saga', 'max_iter': 4000, 'C': 0.05}
Best: 0.853293 using {'penalty': 'l2', 'solver': 'saga', 'max_iter': 3000, 'C': 0.03}
```

In the end the grid search method outputs the combination that achieved the highest score. Using this strategy was possible to improve the final accuracy score to 86%.

It is worth mentioning that with more processing capacity it would be possible to test more parameters combinations, thus further increasing the final score, but to this application it would make no difference as the Human Resources department only wants to have a hunch on possible employees that might leave.

Results

Model Evaluation and Validation

When developing models it is important to learn from the data, but at the same time is really important to keep certain level of generalization, as the model will eventually be submitted to data never seen by it. That is why a model presenting 100% accuracy is considered a bad model, as it will probably be overfitting the training dataset.

The final model presented an accuracy score of 85% in the training set and 86% in the test set, this is considered a good score since the model retained enough information from training to predict with good accuracy and yet kept a good level of generalization towards new data.

The concept of separating the test data from the training data is exactly to avoid overfitting during learning, as the model will always be tested with unseen data, this is a sanity check essential for building consistent models.

Justification

Currently Human Resources department has no tool to monitor neither warnings when an employee decides to leave the company, normally they are caught off guard and don't have much room to maneuver. With any kind of signalling tool Human Resources could get ready in advance, offering alternatives to the employee.

That is why a model, specially one with 86% of accuracy, would be a most welcomed tool, it would give them for the first time the upper hand on employee's attrition cases. Besides, they will be able to develop specific strategies as they could focus their efforts on potential turnover cases instead of having to develop a broad and too generalized talent retention strategy.

Conclusion

Free-Form Visualization

Using the final model a series of predictions were made pointing potential new cases of employee attrition (img. 5).

The model make some wrong predictions, but in general has a good accuracy rate. This predictions are of great help to the Human Resources department as now they can come up with a specie strategy to each employee flagged as potential turnover, number 1 in the predictions column.

img. 5 Model predictions

	Prediction	Test Data
0	1	1
1	1	1
2	0	1
3	1	1
4	0	0
5	0	0
6	1	1
7	0	0
8	1	1
9	1	1
10	0	0
11	1	1
12	0	0
13	0	0
14	0	0

Reflection

The end-to-end solution of this problem can be summarized in the following stages:

1. Define the initial problem and expected answers;
2. Gather data and consolidate in a single dataset;
3. Preprocess the dataset in order to regularize and standardize it;
4. Analyze the data using classical statistical tools;
5. Assess which features are most important;

6. Balance the data based on the target feature;
7. Divide the preprocessed dataset into a train and test sets;
8. Train a model using the train set;
9. Evaluate the metric score of the model;
10. Enhance the model score by fine tuning the parameters;
11. Evaluate final model using the test set;
12. Deploy final model as a Minimum Viable Product.

During the exploratory data analysis some interesting insights could be found, proving that classical statistical analysis are a very important step when developing prediction models.

The fine tuning of the model requires a heavy processing capacity, in order to test different parameters combinations it demands a lot from the hardware. Proper equipments like GPUs would make it faster, however using normal hardwares as personal computers take hours to complete, proving to be a tiresome step.

The deployment of the final model as Minimum Viable Product is a great step towards creating a product that could be used as a general indicator of employee attrition, considering the model would perform with high levels of accuracy once deployed.

Improvement

The original dataset presented a high number of cases of employees turnover when the employee are 60 years or more, however there were few cases of employee attrition for younger talents, this is a problem as the final model will probably focus more on higher ages employees. In the long run this could also cause problems to the company as policies to retain younger talents would be relented to second place, causing a lost of talents that would probably fulfill higher positions in the future.

A good approach would be to gather more information about younger age turnovers or even divide the original dataset so two models could be created, one for elderly employees and other to younger ones. This would probably present different insights for the different age groups, helping to reach a broader group of employees.

The final model has a lot of potential for improvement, either by gathering more information, testing more parameters combinations or by creating age clusters, regardless the type of enhancement its implementation should be weighted against the real necessity of the company, as some improvements proposals could cost more than they would generate in value. That is way it is always important to understand the final user demand and necessity.

References

Porter, J. (2011). Attract and retain top talent. Strategic Finance. 92(12), 56-60. doi:2373925461

Kashyap, Bhuva; Kriti, Srivastava (2018). Comparative Study of the Machine Learning Techniques for Predicting the Employee Attrition. IJRAR July 2018 , Volume 5, Issue 3.

Frye, Alex; Boomhower, Christopher; Smith, Michael; Vitovsky, Lindsay; and Fabricant, Stacey (2018) "Employee Attrition: What Makes an Employee Quit?," SMU Data Science Review: Vol. 1 : No. 1 , Article 9.