Machine Learning Engineer Nanodegree

Capstone Project

Topic: Attrition prediction with machine learning approach in human resource management

Ivan Chen
January 4, 2018

I. Definition

Project Overview

In human resource management, HR professionals take the responsibility in designing suitable systems such as work culture, compensation or promotion system to help the company or organization to retain top employees. Attrition in human resources refers to the gradual loss in labor over time.

In general, average attrition rate compared with other companies in same business is acceptable but relatively high attrition may cause trouble for companies. A major problem in high attrition rate is high cost to an organization; for example, human resources team may need to start hiring processes, providing new hire training. Besides, to the organization, high attrition rate may lead to lose knowledge and experience; it may impact daily or business operation. Usually human resources teams take attrition rates as factors into their department budgets to account for potential losses in productivity and the costs with replacing employees.

Machine learning techniques were used to predict employee turnover in prior paper [1]. The dataset used in this project was downloaded from kaggle, (https://www.kaggle.com/ludobenistant/hr-analytics-1/data) which was provided by Ludovic benistant for HR Analytic study and practice.

Problem Statement

The attrition rate is determined by dividing the number of employees who left the company or organization during a specific period by the number of employees during the same period. Attrition rate can be computed by monthly, quarterly or annual periods. Consistent rate of attrition is treat as the norm for a company but high attrition rate may induce trouble to a company. To predict and classify high attrition risk of employees are critical for human resources members. With the classified result human resources teams may evaluate the causes of attrition and find solutions.

The attrition prediction problem would be a binary classification problem and the goal of the project is to generate a relatively accurate model to predict attrition or stay of employees based on collected employees data. In the study, a solution was proposed and to be implemented. Benchmark model, evaluation metrics, key feature importance observation will be considered and discussed.

The input is a datasest of employees' information including the attrition status which is either 1 (left) or 0 (still stay in company) in CSV format The goal of the model is predicting whether an employee will leave or stay in the company.

Metrics

For this attrition prediction, F-beta score is appropriate metric for this kind of binary classification problem. F-beta score is proposed to be the main metric and accuracy as supporting metric.

Accuracy measures how often the classifier makes the correct prediction. It’s the ratio of the number of correct predictions to the total number of test data points. That is, accuracy = (True Positives + True Negatives) / Total number of data points.

“F-beta score” is a metric that considers both precision and recall. The formula was described below:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

Illustration 1: Metrics: F-score

In the formula of F-beta score, the precision is the ability of the classifier not to label as positive a sample that is negative. For example, in a spam email classification application, it can tell what proportion of messages was classified as spam, actually was spam. That is,

Precision = True Positives/(True Positives + False Positives)

The recall is the ability of the classifier to find all the positive samples. In a spam email classification application, it can tell what proportion of messages that actually was spam that was classified by us as spam. That is,

Recall = True Positives/(True Positives + False Negatives)

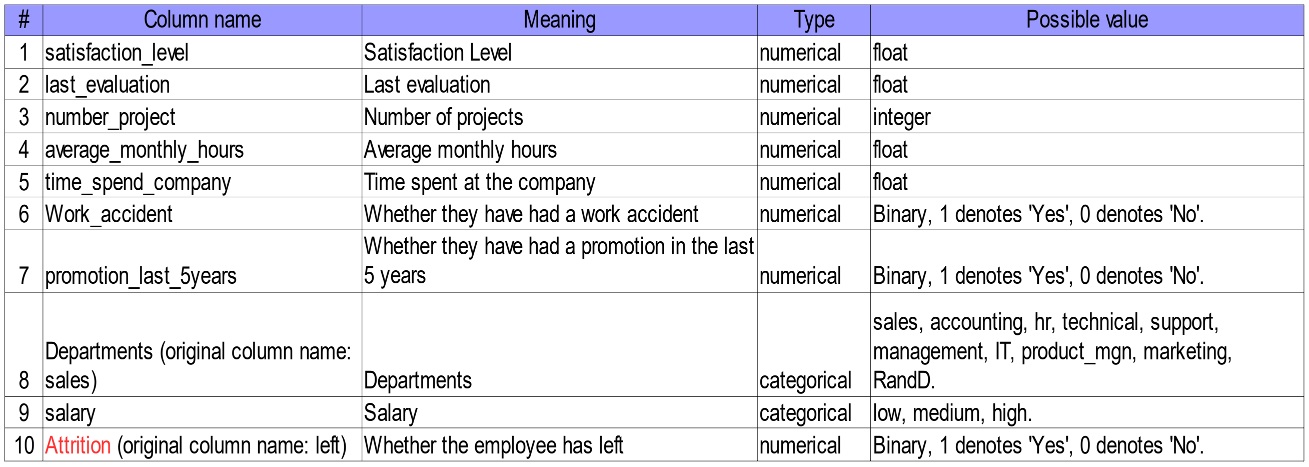
The F-beta score is a weighted harmonic mean of the precision and recall. User for different interest of application can adjust the weight beta. If the value of beta equals 1 means recall and precision are equally important.

For the attrition prediction problem, I thought precision is more important than recall. If an employee were classified as high risk of attrition candidate, it's better to be a real attrition because HR professionals may take some action based on the prediction result, for example, providing compensation program if the employees are valuable or preventing the employee's improper operations that hurt the company before leaving. So I propose to adjust the beta to smaller than 1 because the model requires high precision.

II. Analysis

Data Exploration

The dataset used in this project was downloaded from kaggle which was provided by Ludovic benistant for HR Analytic study and practice. The dataset consists total 14999 rows with 10 columns. The last one “attrition” column renamed by me from “left” for this study was used as target label of the prediction. Other columns provide related information of the employee such as average monthly working hours, number of projects.



#	Column name	Meaning	Type	Possible value
1	satisfaction_level	Satisfaction Level	numerical	float
2	last_evaluation	Last evaluation	numerical	float
3	number_project	Number of projects	numerical	integer
4	average_monthly_hours	Average monthly hours	numerical	float
5	time_spend_company	Time spent at the company	numerical	float
6	Work_accident	Whether they have had a work accident	numerical	Binary, 1 denotes 'Yes', 0 denotes 'No'.
7	promotion_last_5years	Whether they have had a promotion in the last 5 years	numerical	Binary, 1 denotes 'Yes', 0 denotes 'No'.
8	Departments (original column name: sales)	Departments	categorical	sales, accounting, hr, technical, support, management, IT, product_mgn, marketing, RandD.
9	salary	Salary	categorical	low, medium, high.
10	Attrition (original column name: left)	Whether the employee has left	numerical	Binary, 1 denotes 'Yes', 0 denotes 'No'.

Illustration 2: Feature list

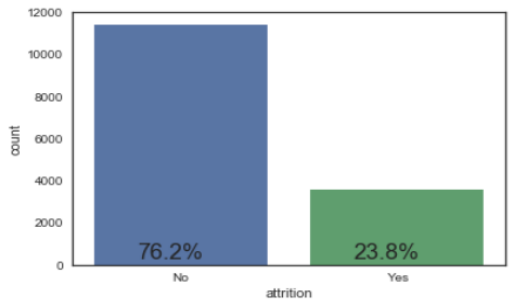
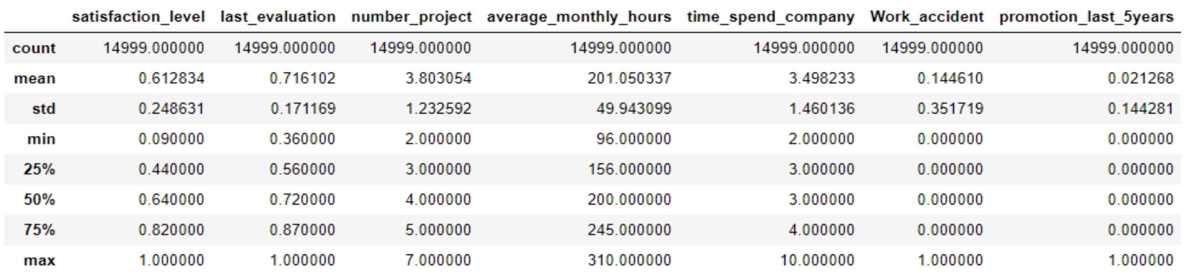


Illustration 3: Attrition in dataset

Although the number of columns is not many but they provide important work related information of employees. Besides, the distribution of target variable “attrition” (76.2% is No, “23.8% is Yes) is unbalanced but I think it is not skewed, so the dataset is suitable for this attrition prediction problem.

The dataset was checked that there is no missing data, no NaN values in all rows and columns.

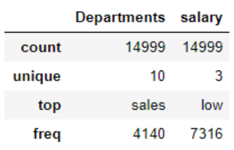
For those numerical features, the summary statistics are list as follows. The “Work_accident” and “promotion_last_5years” are binary values (either 0 or 1). The range of “satisfaction_level” and “last_evaluation” is 0 to 1 which may have been normalized. From the statistical data, I think the features “number_project” and “average_monthly_hours” has no outlier. For the “time_spend_company”, it might have skewed because the 25% and 50% both are 3.0 and close to minimum value which is 2.0; the value of 75% is close (4.0) to mean (3.49). We may see the distribution plotting in next section.



	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000

Illustration 4: Numerical features summary statistics

As for categorical features, the "Departments" has 10 types, and "salary" has 3 types. From the top frequency value, the data should have no big skew problem for training models.

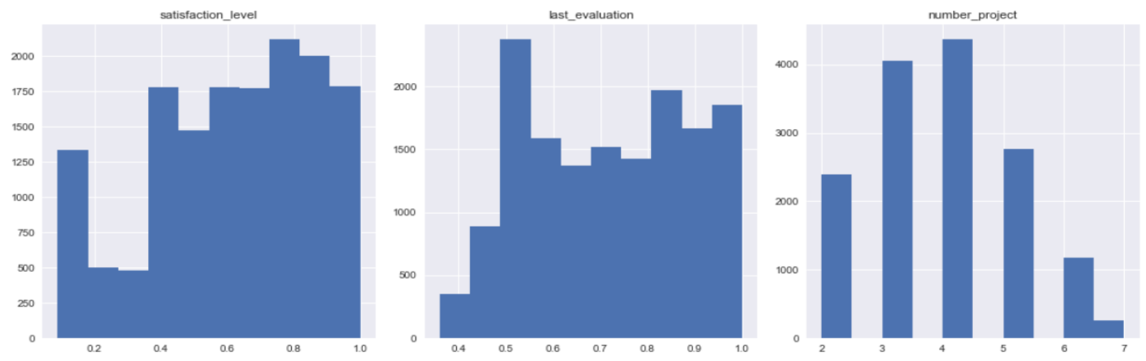


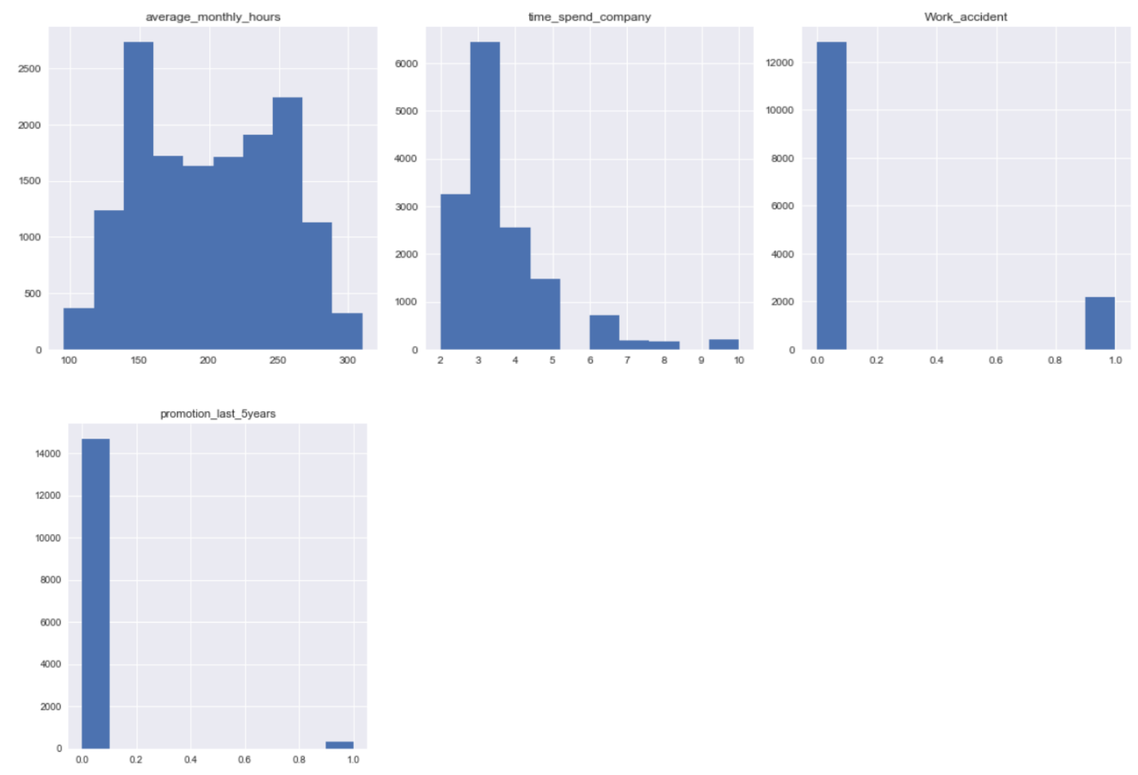
	Departments	salary
count	14999	14999
unique	10	3
top	sales	low
freq	4140	7316

Illustration 5: Categorical features summary statistics

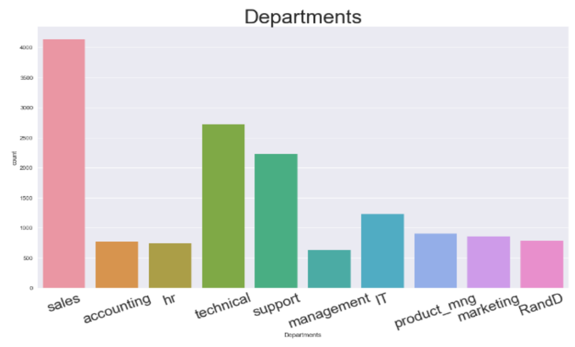
Exploratory Visualization

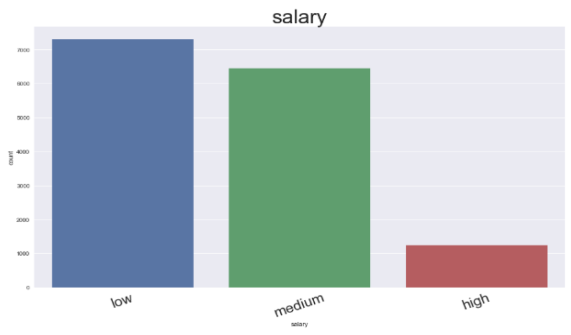
There are seven columns are numerical types, the distribution were illustrated as follows. Features "Work_accident" and "promotion_last_5years" are binary value, either 1 (denotes "Yes") or 0 (denotes "No). The "time_spend_company" has skewed and may need to be pre-processed before training models.





Two features are categorical, the distribution and possible values are illustrated below.





From the correlation map of features below we may see the feature “average_monthly_hour” has relative high correlation with “number_project”. It denotes that the more projects an employee evolves the higher average monthly working hours he/she takes. The two features may related to employee's working loading also.

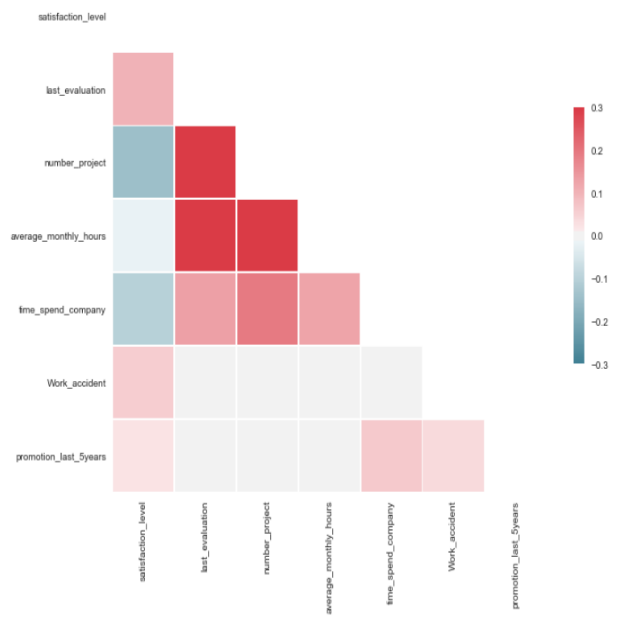


Illustration 6: Features correlation

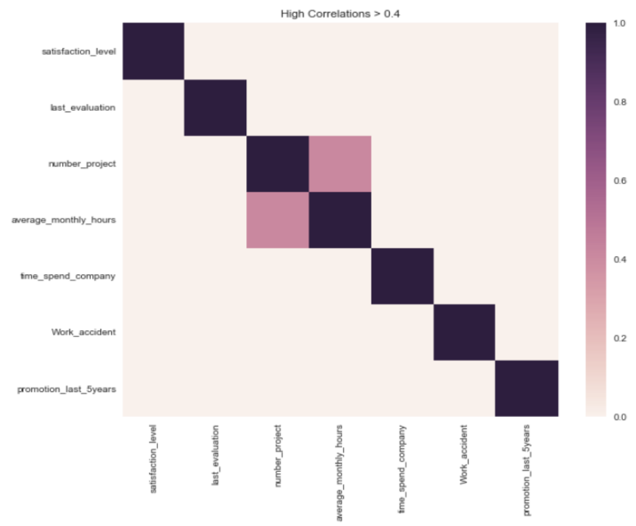


Illustration 7: High correlations > 0.4

Algorithms and Techniques

The workflow of the solution for this attrition prediction consists the following main steps and described below. For this attrition predicting, three supervised learning models to create a training and prediction pipeline were adopted.

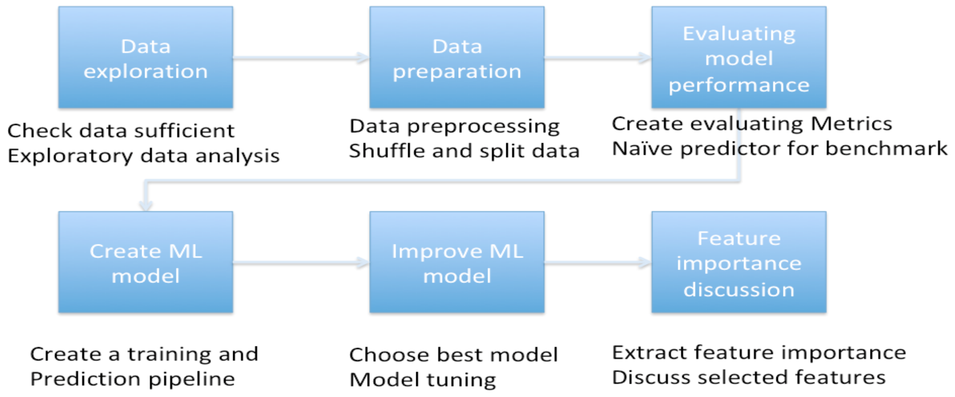


Illustration 8: Workflow of proposed solution

1. Data exploration

First, I will check the dataset whether it is sufficient or suitable for the attrition study or not, then perform exploratory data analysis, such as exploring how may rows in the dataset, what is the attrition ration in the raw data, which I can judge is there large skew of the target label (attrition)? After feature exploration, I may know the types of features, what types are numerical (or binary), what types are categorical they are for later data preprocessing.

2. Data preparation

After data exploration, data preprocessing for different kind of types is needed, such as transferring skewed continuous features, normalization numerical features, using one-hot encoding scheme to convert categorical features to numerical dummy variables. After data was preprocessed properly, then I may shuffle and split data for later training and prediction model creation.

3. Evaluating model performance

In this attrition prediction project, F-beta score metric was chosen to measure the performance of model. Meanwhile, human resources member may care about more precision than recall, a smaller than 1 beta may suitable to get high precision model. Because there is no previous version of model for benchmark, I propose to generate a naive predictor, which always predicts attrition or always non-attrition as base model without any intelligence for comparison.

4. Create machine-learning model

In this attrition perdition, I will adopt three supervised learning models creating a training and prediction pipeline, which are Support Vector Machine (SVM), Decision Trees and AdaBoost ensemble learning. They both have advantages and disadvantages and I think intuitionally the AdaBoost ensemble leaning or Decision Trees may have better performance in the study, but the measurement metrics may tell us the result finally.

The three methods were described briefly as follows.

(1) Support vector machine

Support vector machine are a supervised learning method used for both classification or regression. In this attrition prediction study, the support vector “classification” was adopted.

Given a set of labeled training data, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by the largest distance as possible. We may say SVM algorithm try to find an optimized hyperplan (separation) which has the largest distance to nearest training-data point of any class. That is the largest margin will lead to lowest generalization error of the classifier.

References: https://en.wikipedia.org/wiki/Support_vector_machine

(2) Decision Trees (Dts)

Decision-trees are a supervised learning method used for both classification or regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. In this attrition prediction study, the decision tree “classification” was adopted.

A common strategy of decision tree learning can be described as below:

a. Splitting the source set into subsets based on an attribute value test.

b. Then the process is repeated on each derived subset, the recursive manner is called recursive partitioning.

c. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions.

References: https://en.wikipedia.org/wiki/Decision_tree_learning

(3) AdaBoost

AdaBoost is a machine-learning method short for “Adaptive Boosting” which is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers.

At the beginning, AdaBoost build a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. At each iteration t, a weak learner is selected and assigned a coefficient such that the sum training error of the resulting boost classifier in iteration t is minimized.

References: https://en.wikipedia.org/wiki/AdaBoost

5. Improve machine-learning model

Based on previous evaluation of performance, I may choose the best model and perform model tuning for optimization. If the training time is not a big issue, then I may choose the model with better F-beta score as best model. By using grid search approach with various values of hyper parameters to fine tune the chosen model can get final optimal model.

K-Fold Cross Validation approach was used for verifying the robustness of the final model, that is once the best model was decided, the model can be fine-tuned with grid search cross validation (GridSearchCV) approach with important parameters. With this grid search approach, data fitting on a dataset all the possible combinations of parameter values are evaluated and then the best combination is retained.

K-Fold cross validation approach was described as follows. In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation.

K-Fold cross validation is suitable for verifying the robustness of the final model because all observations are used for both training and validation, and each observation is used for validation exactly once. In this study, 5-fold cross-validation was used. Because the model with decision-tree methodology is the best, “**max_depth**” of decision-tree which is an important feature to control the size of the trees was used to fine-tune the model.

References: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation

6. Feature importance discussion

In the final stage, I will explore the feature importance and their relevance with target label (attrition). With the feature importance observation, if the training time is a critical concern, then new model with these top features (say top five) may have a little lower score than the model with completed features but training time may significantly reduced. Besides, with these top important features observation, human resources teams may consider to provide compensation programs to valuable employees or take prevention actions to avoid employees’ improper operation to the company before leaving.

Benchmark

In this project, I propose to generate a naive predictor, which always predicts attrition (value 1) or always non-attrition (value 0) as base model without any intelligence for comparison. F-score measurement can be adopted to measure the result. With this generated cost-effective naïve model, we may have base model to compare performance once the proposed solution model is ready.

III. Methodology

Data Preprocessing

After data exploration, data preprocessing for different kind of types is needed, such as transferring skewed continuous features, normalization numerical features, using one-hot encoding scheme to convert categorical features to numerical dummy variables.

In this dataset, the feature “time_spend_company” is a little skewed. A common method to avoid outliers to negatively affect the performance of a learning algorithms is applying logarithmic transformation on the data. We can see the range of distribution was narrowed in the below charts. The data was deskewed before passing to models.

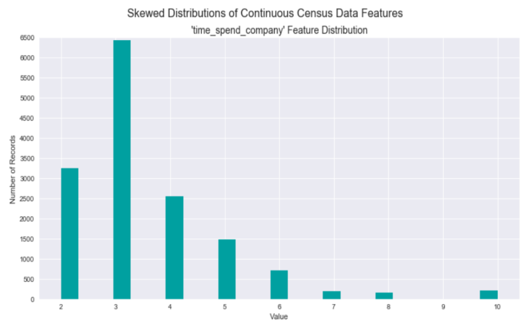


Illustration 9: Skewed distributions: "time_spend_company"

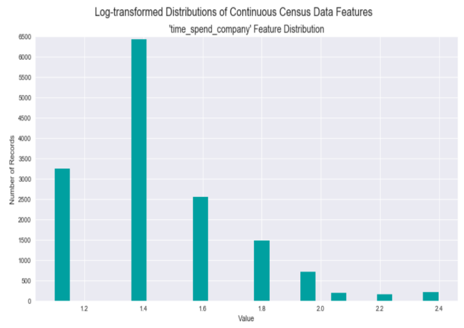


Illustration 10: Deskewed distribution after performing "Log-transformed"

To ensure that each feature is treated equally when applying supervised learners, scaling was applied on all numerical features including 'satisfaction_level', 'last_evaluation', 'number_project', 'average_monthly_hours' and 'time_spend_company'. Two numerical features 'Work_accident', 'promotion_last_5years' are numerical but they are binary values, either 1 or 0 that denotes “Yes” and “No”, so they are no needed to apply scaling. Applying a scaling to the data does not change the shape of each feature's distribution. The target variable “attrition” is either 1 or 0, it is no need to preprocess.

From the data exploration above, we can see there are several features for each record that are non-numeric. Because learning algorithms expect input to be numeric, one-hot encoding scheme was applied to covert those non-numeric features (called categorical variables) to numeric. One-hot encoding creates a *"dummy"* variable for each possible category of each non-numeric feature. For example, the “salary” feature has three possible entries: low, medium and high, three dummy variables “salary_low”, “salary_medium” and “salary_high” were created after the conversion.

After data was preprocessed properly, then I may shuffle and split data for later training and prediction model creation.

Implementation

In this attrition prediction implementation **Scikit-learn (sklearn)** machine-learning library for Python was used for data processing and machine-learning model creating.

In this dataset, the feature “time_spend_company” is a little skewed. Log-transform provided by numpy module was applied to it. The data was deskewed before passing to models to avoid outliers to negatively affect the performance of a learning algorithms.

```
# Log-transform the skewed features
features_log_transformed = pd.DataFrame(data = features_raw)
features_log_transformed[skewed] = features_raw[skewed].apply(lambda x: np.log(x + 1))
```

Typically, learning algorithms expect input to be numeric, which requires that categorical variables be converted to numerical. get_dummies() in Pandas module can be used for this kind of on-hot encode conversion.

```
# One-hot encode the 'features_log_minmax_transform' data using pandas.get_dummies()
features_final = pd.get_dummies( features_log_minmax_transform )
```

In this attrition perdition, I have adopted three supervised learning models creating a training and prediction pipeline, which are Support Vector Machine (SVM), Decision Trees and AdaBoost ensemble learning. They both have advantages and disadvantages and I think intuitionally the AdaBoost ensemble leaning or Decision Trees may have better performance in the study, but the measurement metrics may tell us the result finally.
The three supervised learning models were described briefly as follows:

(1) Support vector machines (SVMs)

Strengths: . Effective in high dimensional spaces. . Still effective in cases where number of dimensions is greater than the number of samples. . Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. . Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.
Weaknesses:
. If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and . regularization term is crucial.
. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

Since the number of features are small and there may be a clear margin of separation, SVC (Support vector classifier) can be a good candidate to define a clear boundary.

(2) Decision Trees
Strengths:
. Simple to understand and to interpret. Trees can be visualized.
. Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
. The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
. Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable. See algorithms for more information. . Able to handle multi-output problems.
Weakness:
. Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
. Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble. The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

Since the number of features are small and easy to understand, few data preparation is needed, I think Decision Tree classification is suitable and easily to implement. DTs (Decision Trees) can be a good candidate to classify attrition.

(3) AdaBoost Ensemble learning

Strength: (Reference: http://user.ceng.metu.edu.tr/~tcan/ceng734_f1112/Schedule/adaboost.pdf)

. Very simple to implement

. Does feature selection resulting in relatively simple classifier . Fairly good generalization

Weakness: (Reference: http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost#Advantages)

AdaBoost can be sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms.

Since the dataset is large and AdaBoost ensemble method is fast, it will be able to perform multiple training iterations to maximize accuracy.

To properly evaluate the performance of each model, a training and predicting pipeline was created that allows effectively train models using various sizes of training data and perform predictions on the testing data. **SVM, tree and ensemble packages** were used to implement the main pipeline flow. The main flow of training and predicting pipeline was shown as below.

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier

clf_A = DecisionTreeClassifier( random_state = 0 )
clf_B = SVC()
clf_C = AdaBoostClassifier()

# Calculate the number of samples for 1%, 10%, and 100% of the training data
samples_100 = len(y_train)
samples_10 = samples_100/ 10
samples_1 = samples_100/100

# Collect results on the learners
results = {}
for clf in [clf_A, clf_B, clf_C]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    print clf_name, "..."
    for i, samples in enumerate([samples_1, samples_10, samples_100 ]):
        results[clf_name][i] = \
        train_predict(clf, samples, X_train, y_train, X_test, y_test)

# Run metrics visualization for the three supervised learning models chosen
vs.evaluate(results, accuracy, fscore)
```

F-beta score function was adopted as major metric of performance and beta value 0.5 was set to the metric function because in the attrition prediction precision is supposed more important than recall.

Metrics packages with F-beta score, accuracy score in sklearn were used for the implementation as below:

```
from sklearn.metrics import fbeta_score, accuracy_score

fbeta_score(y_test, predictions, beta = 0.5)
fbeta_score(y_test, best_predictions, beta = 0.5))
```

Refinement

Based on the performance metric above, basically the model with highest F-score can be chosen as best model. Other factor such as training time can be also considered when choosing best model.

K-Fold Cross Validation approach was used for verifying the robustness of the final model, that is once the best model was decided, the model can be fine-tuned with grid search cross validation (GridSearchCV) approach with important parameters. With this grid search approach, data fitting on a dataset all the possible combinations of parameter values are evaluated and then the best combination is retained. In this case five folds were used.

In this case, the model with decision-tree methodology is the best, "**max_depth**" of decision-tree which is an important feature to control the size of the trees was used to fine-tune the model. "model_selection" module in sklearn was used for Grid search cross-validation.

```
# K-Fold Cross Validation, K=5 in the study
#  Import 'GridSearchCV', 'make_scorer', and any other necessary libraries
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
parameters = {'max_depth': range(1, 10)}

#  Make an fbeta_score scoring object using make_scorer()
scorer = make_scorer( fbeta_score, beta=0.5 )

# Perform grid search on the classifier using 'scorer' as the scoring method using GridSearchCV()
nCV = 5
grid_obj = GridSearchCV( clf, parameters, scorer, cv= nCV )

# Fit the grid search object to the training data and find the optimal parameters using fit()
grid_fit = grid_obj.fit( X_train, y_train.values.ravel())

# Get the estimator
best_clf = grid_fit.best_estimator_

# Make predictions using the unoptimized and model
predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)
```

After model refinement, I will also train it on the same training set with only to top important features whose accumulated normalized weighting is larger than 50% which means these most important features contribute more than half of the importance of all features present in the data. We can reduce the feature space and simplify the information required for the model to learn. We may also expect that the training and prediction time is much lower. In this study, top three feature_importances of model can be extract to build a new model with reduced data.

```
# Import a supervised learning model that has 'feature_importances_'
# Train the supervised model on the training set using .fit(X_train, y_train)
model = AdaBoostClassifier()
model.fit(X_train, y_train)

# Extract the feature importances using .feature_importances_
importances = model.feature_importances_

# Import functionality for cloning a model
from sklearn.base import clone
nFeatures = 3
# Reduce the feature space
X_train_reduced = X_train[X_train.columns.values[(np.argsort(importances)[::-1])[:nFeatures]]]
```

```
X_test_reduced = X_test[X_test.columns.values[(np.argsort(importances)[::-1])[:nFeatures]]]


# Train on the "best" model found from grid search earlier
clf = (clone(best_clf)).fit(X_train_reduced, y_train)


# Make new predictions
reduced_predictions = clf.predict(X_test_reduced)
```

IV. Results

Model Evaluation and Validation

In the model evaluation, the number of records was selected with 1%, 10% and 100% of the training data respectively, that is 119, 1199 and 11999 rows. And three models (AdaBoostClassifier, Support Vector Classifier and Decision-tree classifier) were adopted to train on the testing data and predict on the testing data respectively with default settings for each model.
The performance metrics and result for these tree supervised learning models were as follows.

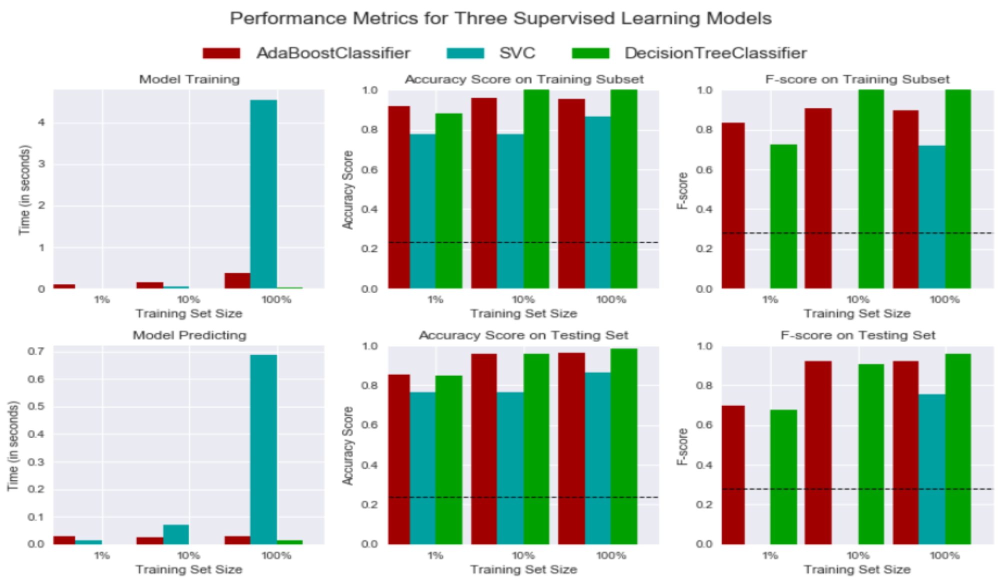


Illustration 11: Performance Metrics for models

(1) 1% of the training data:

Samples_119	Training time	Accuracy	F-score
AdaBoost	0.09	0.85	0.7
SVC	0	0.77	0
Decision-tree	0	0.85	0.68

(2) 10% of the training data:

Samples_1199	Training time	Accuracy	F-score
AdaBoost	0.15	0.96	0.92

SVC	0.05	0.77	0
Decision-tree	0.01	0.96	0.91

(3) 100% of the training data:

Samples_11999	Training time	Accuracy	**F-score**
AdaBoost	0.39	0.96	0.92
SVC	4.55	0.87	0.76
Decision-tree	0.03	0.98	**0.96**

Based on the result of F-score on the testing when 100% of the training data is used, both the F-score and accuracy of Decision-tree methodology are higher than other two methods. Besides, from the training time, Decision-tree (0.03 sec) is much faster than SVC (4.55 sec) and a slightly faster than AdaBoost (0.39 sec), considering the prediction and training time, Decision-tree classifier seems be the best model.

Based on the performance metric above, decision-tree classifier is the best model, then the model can be fine-tuned with grid search cross validation (GridSearchCV) approach with important parameters “max_depth”. In this case, “max_depth: 8” of decision-tree is the best parameter to generate the optimized model, and the F-score of optimized model is **0.973** which is better than unoptimized model which is 0.956. From the table below we may see in the max_depth 8, the mean of test score of five folds is the highest whose test_score of five splits are 0.97, 0,97, 0.976, 0.971 and 0.975. The standard deviation of testing score at Max_depth 8 is only 0.00268, it was slower than others (see the chart below), so I think we may say the robustness of the final model is good.

param_max_depth	**mean_test_score**	**std_test_score**	**split0_test_score**	**split1_test_score**	**split2_test_score**	**split3_test_score**	**split4_test_score**
1	0.625104	0.008906	0.619181	0.629945	0.629338	0.610976	0.636086
2	0.721	0.010045	0.730725	0.729537	0.704182	0.715123	0.725434
3	0.889117	0.012375	0.884537	0.878689	0.898625	0.875584	0.908156
4	0.937788	0.006624	0.929282	0.931645	0.945848	0.937724	0.944444
5	0.954539	0.007827	0.946724	0.946466	0.966448	0.952639	0.960421
6	0.962354	0.005964	0.95945	0.952214	0.968452	0.964207	0.967447
7	0.973078	0.00291	0.968093	0.971725	0.976419	0.974576	0.974576
8	**0.973117**	**0.002682**	0.970909	0.970149	0.976821	0.971918	0.975789
9	0.968466	0.002652	0.967101	0.964512	0.972122	0.968093	0.970503

Illustration 12: Scores of five folds cross validation

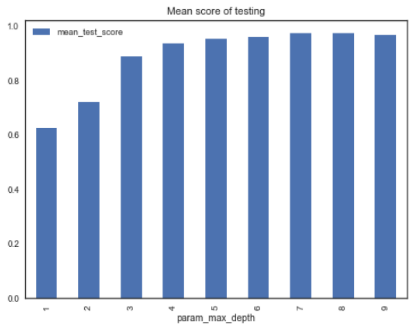


Illustration 14: Mean score of testing

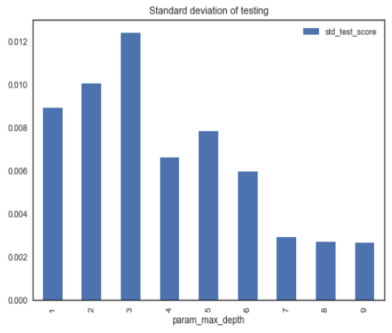
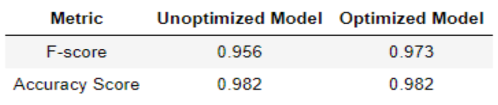


Illustration 13: Standard deviation of testing



Metric	Unoptimized Model	Optimized Model
F-score	0.956	0.973
Accuracy Score	0.982	0.982

Illustration 15: F-score of optimized model

Feature selection for model with reduced data

Top five important features illustrated below are “satisfaction_level”, “average_monthly_hours”, “number_project”, “time_spend_company” and “last_evaluation”. And the accumulated normalized weights of *top three* only is larger than 0.5 which means these three most important features contribute more than half of the importance of all features present in the data.

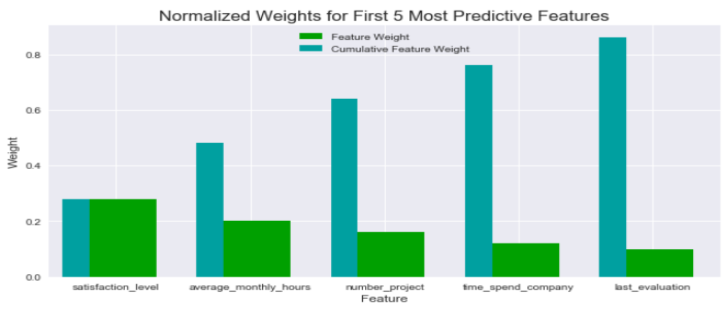
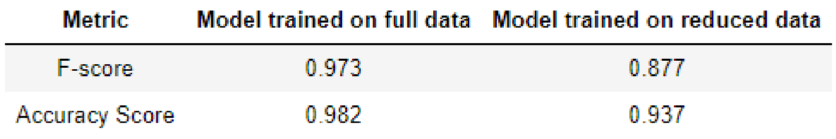


Illustration 16: Normalized weights for top 5 most predictive features

The F-score of model trained on full data and *reduced data* (*top three* important features) was 0.973 and 0.877.



Metric	Model trained on full data	Model trained on reduced data
F-score	0.973	0.877
Accuracy Score	0.982	0.937

Illustration 17: F-score of model trained on full data vs. reduced data

Justification

The F-score of optimized model is 0.973 and accuracy is 0.982. These scores are better than the unoptimized model whose F-score is 0.956 and accuracy is 0.982. The naive predictor's F-score is 0.28 and accuracy score is 0.238. The optimized model's scores are much better than naive predictor's whose F- score is 0.2809 and accuracy score is 0.2381.

The F-score of model trained on reduced data is 0.877, lower than the full data but I think it is acceptable if the training time is a concern. Besides, we may know these top three important features “satisfaction_level”, “average_monthly_hours”, “number_project” contribute half importance of all features present in the attrition data.

V. Conclusion

Free-Form Visualization

From the F-score metrics, the model with decision tree classifier has great performance of prediction for the attrition dataset. The training time is relatively smaller than other two algorithms. Since the number of features are small and easy to understand, few data preparation is needed, I think Decision Tree classification is suitable for such kind of attrition prediction for human resource management.

From the feature selection for model with reduced data, both F-score and accuracy on the reduced data using only three features are a little lower than those when all features are used. If training time was a factor, I will consider using the reduced data as my training set to make a trade-off between accuracy and training time. From the accumulated normalized weights of features we may know these top three important features “satisfaction_level”, “average_monthly_hours”, “number_project” contribute half importance of all features present in the attrition data. Human resource professionals may consider to provide compensation program to those valuable employee with high attrition risk if they have low satisfaction level and high average monthly working hours or relatively high number of evolved projects.

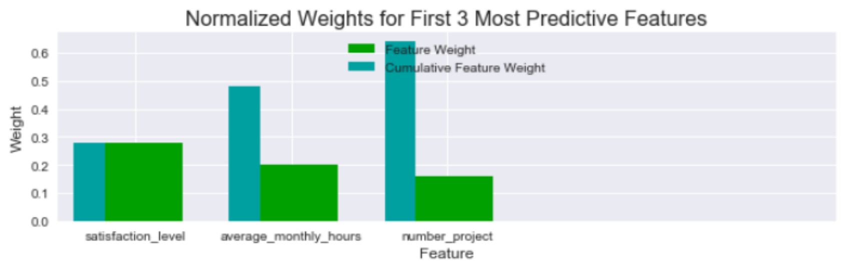


Illustration 18: Accumulated normalized weights for top 3 most predictive features

Reflection

The main process used for this project can be summarized using the following steps: 1. Data exploration, 2. Data preparation 3. Evaluating model performance, 4. Create machine-learning model, 5. Improve machine-learning model and 6. Feature importance discussion.

I think in step 5 “improve machine-learning model” is the most interesting part for me. Intuitionally I thought AdaBoost method will have best F-Score but in fact the decision-tree has best performance from the result of step 4. The F-score is high enough (0.956) which is beyond my expectation, but after refining the model in step 5 even higher F-score (0.973) was got with the five folds of grid search cross-validation approach. Data may tell us the fact, the truth and it may beyond human's intuition. It is the interesting part of this machine learning study work for me.

Improvement

In this attrition study, the final F-score of best model Decision-tree is high with input dataset from kaggle but I think in real-world the human resource related data maybe more complicated or even non-structured, there should rooms to improve for attrition prediction.

From the top five or top three important features, we may know the key factors are “satisfaction_level” and work loading (“average_monthly_hours” and “number_project”), if we have more detailed information about these features, we may generate more features to generate accuracy models. For example, “satisfaction_level” may be extended to the company, satisfaction to the work, satisfaction to the facility, satisfaction to the salary or compensation; work loading may be extended to such as involved project type, cooperation department, cooperation colleague. With these extended features, human resource team may consider to provide compensation program or improvement to those valuable employees to resolve their concerns. If human resource team can collect the attrition data before and after deploy these compensation program of improvement, then attrition perdition model can be expanded to relate to compensation and improvement program.

Basically the attrition prediction is a supervised binary classification problem, if the number of column is large and hard to interpret by human we may use unsupervised learning method, such as K-maen clustering, Principal component analysis (PCA) to find the inside information for attrition prediction for human resource management.

VI. References

[1] Rohit Punnoose, Pankaj Ajit , BITS Pilani, “Prediction of Employee Turnover in Organizations using Machine Learning Algorithms.” https://pdfs.semanticscholar.org/fa49/19810eaee67e851ad13775b78c94217a7908.pdf