Machine Learning Project Report

1. Problem Description

Goal: Predict if a data scientist will look for a job change and evaluate which factors affect his decision.

Use case:

A company that is hiring employee can know among the candidates which one is really looking for a new job, which reduces the cost and time of finding new employees.

Predict which employee is likely to look for a job change, and analyze which factors influent his decision.

2. Data

2.1 Features

Each example in the dataset is a vector of 13 x 1 vector with the following feature

|  |  |  |
| --- | --- | --- |
| **Feature** | **Type, domain** | **Description** |
| **enrollee\_id** | **int64**  **Example:**  [ 8949, 29725, 11561, ...,] | Unique ID for candidate. |
| **city** | **object**  **Example:**  ['city\_103', 'city\_40', 'city\_21', 'city\_115', ...] | City code. |
| **city\_development\_index** | **float64**  **Example:**  [0.92 , 0.776, 0.624, 0.789, ...] | Development index of the city. |
| **gender** | **object**  **Domain:**  ['Male', nan, 'Female', 'Other'] | Gender of the candidate. |
| **relevent\_experience** | **object**  **Domain:**  ['Has relevent experience', 'No relevent experience'] | Whether the candidate has relevent experience. |
| **enrolled\_university** | **object**  **Domain:**  ['no\_enrollment', 'Full time course', nan, 'Part time course'] | Type of University course the candidate enrolled. |
| **education\_level** | **object**  **Domain:**  ['Graduate', 'Masters', 'High School', nan, 'Phd', 'Primary School'] | Education level of the candidate t. |
| **major\_discipline** | **object**  **Domain:**  ['STEM', 'Business Degree', nan, 'Arts', 'Humanities', 'No Major', 'Other'] | Education major discipline of the candidate. |
| **experience** | **object**  **Domain:**  ['>20', '15', '5', '<1', '11', '13', '7', '17', '2', '16', '1', '4', '10', '14', '18', '19', '12', '3', '6', '9', '8', '20', nan] | Candidate’s total experience, measured in years. |
| **company\_size** | **object**  **Domain:**  [nan, '50-99', '<10', '10000+', '5000-9999', '1000-4999', '10/49', '100-500', '500-999'] | Number of employees in current employer's company. |
| **company\_type** | **object**  **Domain:**  [nan, 'Pvt Ltd', 'Funded Startup', 'Early Stage Startup', 'Other', 'Public Sector', 'NGO'] | Type of current employer’s company. |
| **last\_new\_job** | **object**  **Domain:**  ['1', '>4', 'never', '4', '3', '2', nan] | Time interval between previous job and current job  (in years). |
| **training\_hours** | **int64**  **Example:**  [ 36, 47, 83, 52, 8, 24, 18, 46, 123, 32, ...] | Training hours completed. |

Output:

An assigned label of either“yes” or “no” for the question whether the candidate is looking for a new job.

Values: 0 or 1

0 – The candidate is not looking for a new job.

1 – The candidate is looking for a new job.

2.2 Explore

We remove enrollee\_id from the attributes because it is not relevant to the result.

The first noticeable characteristic of data is that it is imbalance. Number of examples in class 0 is about 3 times than the number of examples in class 1. This must be addressed while building the model.



The data can be divide into 3 type of attributes:

- numerical: city\_development\_index, training\_hours

- ordinal categorical: relevent\_experience, enrolled\_university, education\_level, experience, company\_size, last\_new\_job

- nominal categorical: gender, major\_discipline, company\_type

The following attributes have missing values: gender, enrolled university, education\_level, major\_discipline, experience, company\_size, company\_type, last\_new\_job.

2.2.1. Numerical atributes

We plot the correlation matrix of numerical attributes and target column



We can see that among the numerical attributes, city\_developement\_index is the one influence target the most, the higher the city\_development\_index, the less likely that a person will change job.

2.2.2. Ordinal categorical attribute

We count the number of target value correspond to each value of attributes and normalize it to compare the proportion between attributes and obtain the following graphs:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Upon inspecting the above graph we have the following conclusions:

- People who have relevant experience is less likely to change job.

- People who enrolled in university full time is more likely to change job.

- Graduate is the most likely to change job, while people who has PhD or Primary School degree is less likely to change job.

- In general, company size doesn't significantly influence whether a person change job or not.

- The less time between the last job and the current job, the more likely a person will change the job. Most noticeably, people who never have a job are the most likely to change job.

2.2.3. Nominal categorical attributes

|  |  |
| --- | --- |
|  |  |
|  |  |

Similar to ordinal categorical attributes, we obtain the above graphs for nominal categorical and the following observations:

- There are no difference between genders in the decision whether to change job or not.

- In general, people who study Humanities or Art (which is less related to Data Science) are slightly less likely to change job.

- People from Funded Startup are less likely to change job.

2.3 Data Preprocessing

2.3.1 Train, cross validation, test set split

To ensure the proportion of examples of 2 classes is the same for every class, we use stratified split along the target.

2.3.1. Numerical Attributes

We use the standard scaler to normalize data to have mean = 0 and standard deviation = 1.

2.3.2 Ordinal Categorical Attributes

We first transform data to numerical attributes (implementation in custom\_transformer.py) in order to use Logistic Regression and Support Vector Machine.

Next, we fill the missing value in the attributes with the most frequent value. We also tried KNNImputer and IterativeImputer class of sklearn library but the result does not change significantly. Thus, we decide to use the simplest strategy to fill the missing value to accelerate development (since the 2 other methods of imputing missing value takes a significantly longer time to fill data).

Finally, we use again StdScaler to make the data the same scale as the ordinal attributes.

2.3.3 Nominal Categorical Attributes

Similar to ordinal categorical attributes, we also fill missing values with the most frequent value. Moreover, we also add columns to the data that indicate whether the example previously had the missing value. These columns have decent impact on the result, imply missing values play an important role.

Next we use OneHotEncoder to change the value to numerical. Since the attributes are nominal we want to use OneHotEncoder to ensure that there is no ordinal relationship between values.

Finally, we use StdScaler to scale data to the same scale and speed up the learning process of logistic regression and support vector machine as well as prevent svm to be heavily relied on attributes with larger range.

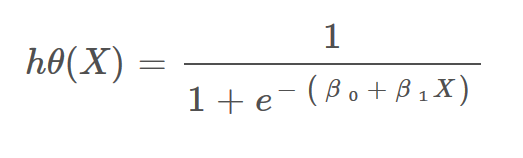
3. Algorithm

3.1 Logistic Regression

In our problem, we will apply logistic regression as a binary classification algorithm to predict whether a person will change job.

3.1.1 Decision Boundary

Logistic regression will learn the vector **β** = [β0, β1, …, βn] correspond to the hypothesis function



Source: <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148#:~:text=For%20logistic%20regression%2C%20the%20Cost,Graph%20of%20logistic%20regression>

The model will predict 1 if h(X) > threshold and 0 otherwise.

3.1.2 Cost function

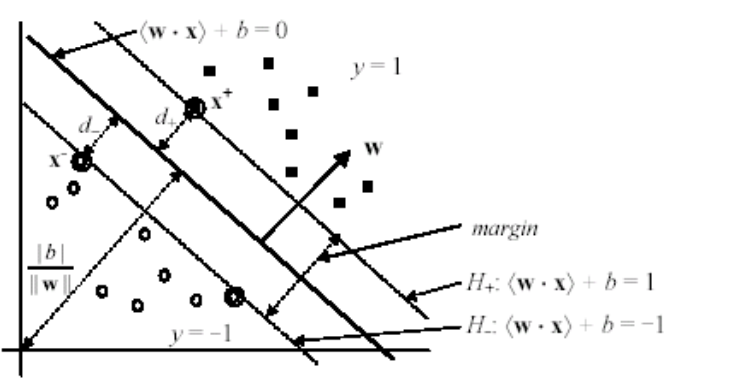
In our problem we apply logistic regression for imbalanced data by using class\_weight = “balanced” parameter and l2 regularization. The algorithm will then try to minimize the following cost function:

* w0, w1 are the weight apply to class 0 and 1 respectively. To fix imbalanced data problem, the algorithm will assign the higher weight to the class with less data in the training set. In sklearn, the weight for each class when apply class\_weight = “balanced” is calculated as: *n\_samples / (n\_classes \* n\_samples\_with\_class)*.
* λ is the regularization coefficient, larger λ will lead to a model that is less overfitted. In scikit-learn implementation, they use a coefficient C which is proportional to the inverse of λ which has the reversed effect (larger C imply higher variance).

Reference: <https://machinelearningmastery.com/cost-sensitive-logistic-regression/>

3.2 Support Vector Machine

Given training vectors , i = 1,... n , and vector , our goal is to find and such that the hypothesis given by the formula below is correct for as many samples as possible:



*(Liu, 2006)*

Margin is the distance between these two (margin) hyperplanes H+ and H-

Formula of the margin:

Primal problem:

subject to:

with is a non-linear mapping (to be discussed later).

Our goal is to maximize the margin, by minimizing ( = )

Since it is certain that the will be misclassified samples (due to noise, error or the nature of dataset), we introduce slack variables :

〈w⋅xi 〉+b ≥ 1− for yi = 1

〈w⋅xi 〉+b ≤ −1+ for yi = -1

The higher the sum , which means the more incorrectly the model predicts, comes with the higher penalty.

The penalty term C controls the strengh of this penalty, and as a result, acts as an inverse regularization parameter ().

The dual problem:

subject to:

Where are called the dual coefficients, and upper-bounded by C.

If we take the derivative wrt and set it equal to zero, we get the following solution, so we can solve for :

Now knowing the we can find the weights w for the maximal margin separating hyperplane:

Kernel and kernel trick:

As mention above, is a non-linear mapping. The reason for using instead of x is to transform the input data space into another higher dimension space so that the transformed one is linearly separable (Recall that SVM formulations require linear separation).

Since we only need the , we use kernel function K<x,z> to calculate that value the those inner-products. This helps us save time finding out , and exact values of . The kernel that we use in this project is Gaussian RBF:

; where

4. Models

4.1 Model tuning:

4.1.1 Logistic Regression model:

We will use Logistic Regression model from sklearn library:

**sklearn.linear\_model.LogisticRegression**

Hyper-parameters choosing:

The following hyperparameter values are selected based on literature survey, and evaluation results using stratified-holdout method. The model with the set of parameters that performs best on cross-validation set will be selected.

* “class\_weight = ‘balanced’”:

Since the classes distribution in the training dataset is imbalanced, we will assign different weight to each class according to its propotion in the training set:

with:

* “penalty = ‘l2’:

We will be using Ridge regularization (Least Square Error):

With *λ* called the regularization parameter. Also, for each sample i, it’s cost function is multiply with the weight of its true class (

* “C = 0.1”:

C is the inverse of regularization parameter *λ.*

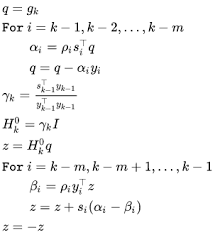
By apply cross-validation, we obtain the table:

|  |  |  |
| --- | --- | --- |
| C value | f1-score on Train set | f1-score on CV set |
| 0.0001 | 0.5757802000895924 | 0.5901639344262296 |
| 0.001 | 0.6114649681528662 | 0.6171284634760706 |
| 0.01 | 0.6123271889400922 | 0.6185480486781368 |
| 0.1 | 0.6133256583681106 | 0.6191275167785234 |
| 1.0 | 0.6132374100719424 | 0.6188679245283019 |
| 10.0 | 0.6133256583681106 | 0.6188679245283019 |
| 100.0 | 0.6133256583681106 | 0.6188679245283019 |

We can see that the model with C equals 0.1 provides the best f1-score result on cross-validation set(0.6191275167785234). Therefore, value 0.1 is selected for C.

* “solver = ‘lbfgs’” (default):

We found out that all descent methods available in this method (‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’) make little to no difference in term of performance on this dataset (all converged in short amount of time, with little difference on accuracy and f1-score). Therefore, we will use the default descent method: ‘lbfgs’.



*LBFGS algorithm - Source:* [*https://en.wikipedia.org/wiki/Limited-memory\_BFGS*](https://en.wikipedia.org/wiki/Limited-memory_BFGS)

4.1.2 SVM Model:

We will use SVC model from sklearn library:

**sklearn.svm.SVC**

Parameters choosing:

* “kernel = ‘rbf’:

We will use the Gaussian RBF Kernel, as chosen in 2.2.

* “C = 200” and “ gamma = 0.0005”:

With C is the inverse of regularization parameter, and gamma is denoted by in the kernel function:

The parameter C trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.

[https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation](https://scikit-learn.org/stable/modules/svm.html%23svm-mathematical-formulation)

Using cross-validation, we obtain the table of f1-score on CV sets, with respect to each gamma and C value:

|  |  |  |  |
| --- | --- | --- | --- |
|  | gamma = 0.0002 | gamma = 0.0005 | gamma = 0.001 |
| C = 100 | 0.6073040623717686 | 0.6092827004219409 | 0.6052974381241858 |
| C = 200 | 0.6053169734151329 | 0.6103004291845493 | 0.5974924340683097 |
| C = 500 | 0.608294930875576 | 0.6024305555555556 | 0.599396291504959 |

We can see that the model with C equals 200, gamma equals 0.0005 provides the best f1-score result on cross-validation set(0.6103004291845493). Therefore, those values are selected.

4.2. Result

We have the evaluation result table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Logistic Regression model | SVM model |
| Train set | Accuracy | **0.77** | **0.80** |
| Recall | **0.74** | **0.77** |
| Precision | **0.52** | **0.57** |
| f1 | **0.61** | **0.66** |
| Cross Validation set | Accuracy | **0.76** | **0.76** |
| Recall | **0.77** | **0.74** |
| Precision | **0.52** | **0.52** |
| f1 | **0.62** | **0.61** |
| Test set | Accuracy | **0.77** | **0.78** |
| Recall | **0.79** | **0.79** |
| Precision | **0.53** | **0.54** |
| f1 | **0.63** | **0.64** |

*(higher values displayed in green color, lower values displayed in red color, two equal values are displayed in blue color)*

5. Discussion and Improvement

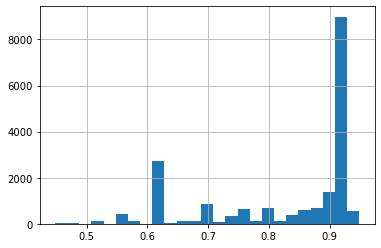
5.1 Discussion

SVM only performs slightly better than logistic regression, however the training time and time taken to predict is much longer. In conclusion, for this problem, logistic regression is preferred.

Compared to other teams’ works on kaggle.com, our models’ performances are on par with other complex models using more advanced libraries and algorithms.

5.2 Improvement

- Collect more data: As shown above when exploring data, city\_development\_index is an important factor to the outcome of the problem. However, with the current distribution of the data (below), it is impractical to perform stratified split along this attributes. In the future, if we can collect more data, we believe that the model can perform better.



- Random Forest Classifier: The dataset has many categorical attributes so we hope that we can apply random forest to try to improve the result.