STATISTICS FOR DATA SCIENCE
GROUP 17: PREDICTIVE MODELLING & ANALYSIS REPORT

Employee Attrition Rate

Abstract

Exploring predictive modelling for accurately classifying employee attrition

Group Members:
Elena Ivon De Jesus Lara
Aahan Kotian
Paul Lefaive
Yongwen Pan
Nayana Sreejith
Lei Ye

April 17, 2023

Introduction

Attrition is the departure of employees from an organization for any reason be it voluntary (ex. resignation or retirement) or involuntary (termination). All human resource departments would want to keep abreast of the rate of attrition occuring at their organization which is calculated by dividing the number of employees leaving the organization by the average number of employees at the organization over a certain period of time.

High rates of attrition can have a devastating impact on any organization. According to Canadian survey The Harris Poll commissioned by Express Employment Professionals, a leading global staffing provider, employee turnover costs companies an average of over $41,000 each year, and some reported costs as high as $100,000 or more per year[1]. These expenses stem from costs to rehire, lost productivity and more. What's problematic is that high levels of attrition are showing an increase in Canada as Canadian businesses look to recover and readjust to the post-pandemic workforce landscape. The Harris Poll also uncovered that 35% of Canadian companies reported an increase in employee attrition, up from 24% in 2021[1].

Data scientists can utilize statistical tools and techniques such as predictive modeling to assist employers and their human resource departments in identifying factors that contribute to overall employee dissatisfaction and turnover. By taking into consideration variables such as an employee's demographics, work history, and workplace sentiment, predictive models can help employers better understand which employees are most likely to leave and strategically address the most significant factors causing this outcome for these employees.

Objectives

The scenario that we as a group have selected to explore involves building a predictive model based on employee attrition data from an unnamed company in order to predict or classify employees as likely to leave the organization or not. This company has been facing a drastic increase in employee attrition, having experienced an increase from 14% to 25% within the last year[2]. In order to accomplish this multiple different models will be trained and tested and output accuracies compared. Ultimately, the model selected would be able to predict attrition with a high level of accuracy that could help to inform and prepare a targeted human resources strategy for increasing employee retention.

Modelling Tools & Techniques

The main programming language that will be used to explore the employee attrition dataset and build our predictive models is python. Several python modules will be utilized for different aspects of the project. For working with the data in the form of a data frame we will be using the pandas and numpy packages. For plotting and graphing our data to visualize our results we will be also using the python module Matplotlib and Seaborn plotting library. For building, training, and testing our predictive models we will leverage both the statsmodels and scikit-learn modules. All coding and analysis will be conducted within a Jupyter notebook.

Data Preparation

Source of Data

The name of the dataset is called “Employee Attrition Rate.” It is an open dataset published to Kaggle.com by Prachi Gopalani in December of 2022. The dataset is free for the public to download and use. The data itself is in CSV file format and a total file size of 19 KB. Download web link for data: https://www.kaggle.com/datasets/prachi13/employeeattritionrate

Data Collection Details

The details provided by the Kaggle user who published the data are quite vague. However, they do state that the employee attrition data originates from an industry leading firm. Based on the data's attributes, the data's origins could be a combination of employee records (human resources) and employee sentiment survey results.

Data Schema & Structure

The dataset consists of 1,470 rows excluding the first row which is a header. Each row represents an employee observation, with each column representing employee attributions:

	Variable	**Type**	**Definition**
Independent Variables	Age	Integer	Age of employee
	Department	String	Department of work
	DistanceFromHome	Integer	Travel distance from home (units unknown)
	Education	Integer	1-below college; 2-college; 3-bachelor; 4-Master; 5-doctorate
	EducationField	String	Education field of study/specialization
	EnvironmentSatisfaction	Integer	1-low; 2-medium; 3-high; 4-very high
	JobSatisfaction	Integer	1-low; 2-medium; 3-high; 4-very high
	MaritalStatus	String	
	MonthlyIncome	Integer	Monthly income in dollars
	NumCompaniesWorked	Integer	Number of previous employers prior to current employer
	Worklife Balance	Integer	1-bad; 2-good; 3-better; 4-best
	YearsAtCompany	Integer	Current years of service with employer
Dependent Variable	Attrition	binary	Employee attrition status, 0-no; 1-yes

Below is a sample of the first 5 rows of the dataset

Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	JobSatisfaction	MaritalStatus	MonthlyIncome	NumCompaniesWorked	WorkLifeBalance	YearsAtCompa
41	Yes	Sales	1	2	Life Sciences	2	4	Single	5993	8	1	6
49	No	Research & Development	8	1	Life Sciences	3	2	Married	5130	1	3	10
37	Yes	Research & Development	2	2	Other	4	3	Single	2090	6	3	0
33	No	Research & Development	3	4	Life Sciences	4	3	Married	2909	1	3	8
27	No	Research & Development	2	1	Medical	1	2	Married	3468	9	3	2
32	No	Research & Development	2	2	Life Sciences	4	4	Single	3068	0	2	7
59	No	Research & Development	3	3	Medical	3	1	Married	2670	4	2	1
30	No	Research & Development	24	1	Life Sciences	4	3	Divorced	2693	1	3	1
38	No	Research & Development	23	3	Life Sciences	4	3	Single	9526	0	3	9

Data Quality & Cleansing

Null Values

Null values should be checked prior to working with any dataset as missing values can cause inaccuracies and skew data analysis results if not excluded or supplemented for . Once the source data CSV file (employee attrition records) were read into a data frame using pandas the “isnull()” method was applied on all attributes in the dataset to identify if null values existed. The resulting output returned false for all attributes meaning that no null values existed in the data.

One-Hot Encoding Categorical Data

The dataset has a number of categorical attributes consisting of string type data. Machine learning models mainly require working with numeric values. Simply converting or reassigning these strings to numeric values is not correct as these models may misinterpret them as ordinal data with some sort of hierarchy when in fact they are unrelated having no order or rank.

Attributes within the dataset that required One-Hot encoding were the columns Department, EducationField, and MaritalStatus. This was accomplished by using the “get_dummies()” pandas function on these attribute columns which essentially creates a new attribute column for each unique category string value and assigns an integer value of 1 or 0 for each new attribute column (1 meaning it belongs to that category). A separate data frame was used for each to generate the one-hot encodings, then concatenated with the main data frame. Below is an example of the One-Hot encodings created for the categorical attributes:

Human Resources	Research & Development	Sales	Human Resources	Life Sciences	Marketing	Medical	Other	Technical Degree	Divorced	Married	Single
0	0	1	0	1	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0	0	0	1
0	1	0	0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...

String to Integer Transformation

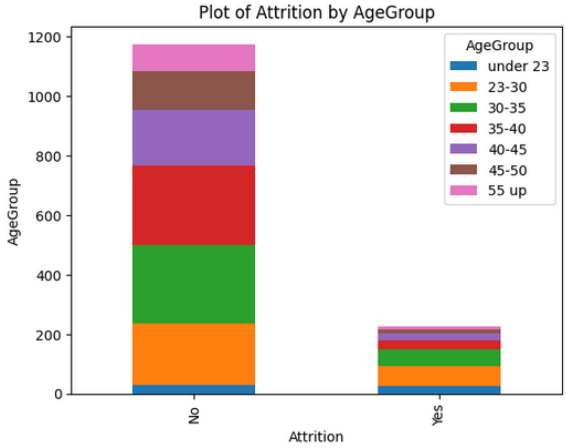
For our dependent variable, Attrition, we had to convert the attribute values from “Yes/No” to integer values 1 or 0 in order to be included in our predictive modeling for the same reasons explained above. This was accomplished by using the “replace()” pandas function which automatically converted the values within the existing data frame for us.

Exploratory Data Analysis

Descriptive Statistics

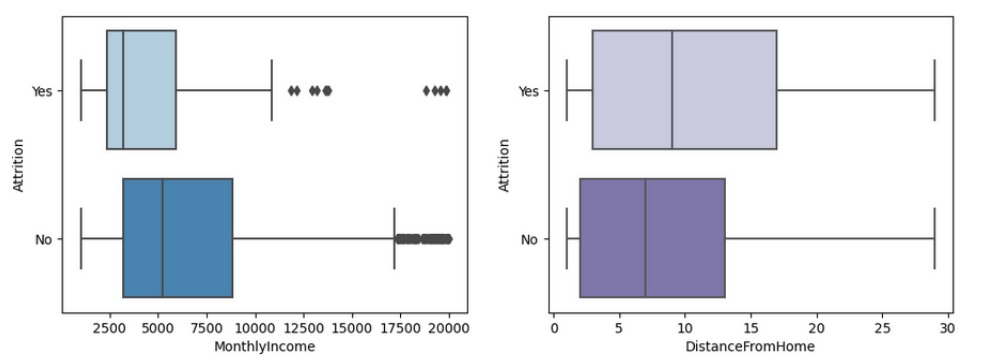
Below is the Descriptive Statistics analysis of the dataset. As we can see, the mean employee age is 36.9 with the minimum being 18 and the maximum being 60. 25% are under the age of 30, 50% are under the age of 36 and 75% are under the age of 43. Knowing this, we can get a gist of what the demographics of the company are and the retention rates among the various age groups as well as what groups the company should strive to promote itself to and retain. Those under the age of 30 are more likely to have a higher churn rate due to comparative lack of experience, expensive training and would thus be harder to retain in the company, while those above the age of 43 might have the similar attrition problems due

to being more likely to retire or under pressure to do so. The results of which can be confirmed by our plot of attrition below:



Distance from home is another aspect of our DS analysis. As you can see, the average distance from the workplace to one's house is 9.1 units in contrast to the minimum being 1 and the maximum number of units being 29. We can also see that out of the distance groups, 25% are less than 2 units away, 50% are less than 7 units away and 75% are less than 14 units away. While we do not know whether these units describe miles or kilometers it is clear to see that employees who live further away from their place of work have a far higher churn rate.

Lastly, the third variable analyzed in the dataset is monthly income. The average monthly income is roughly $6502, with the minimum being $1009 and the maximum being $19,999. 25% of the employees make under $2911, 50% make under $4919 and 75% make under $8379. One can gather that due to the high cost of living, lower income employees would be more motivated to exit the company in order to find positions that would yield a higher pay.



Predictive Modeling

Preparing X & Y Variables

As specified in the Data Preparation section, dummies were created using pandas function "pd.get_dummies()" for the categorical variables where no ordinal relationship existed among the

categories. This was the case for “EducationField”, “Department” and “Marital Status”. Once the dummy variables were created, these were stored in the variable **“x_variables”**. The columns stored as a dataframe in “x_variables” were Age, DistancefromHome, Education, EnvironmentSatisfaction, JobSatisfaction, MonthlyIncome, NumCompaniesWorked, WorkLifeBalance, YearsatCompany and the dummies: Human Resources, Research and Development, Sales, Human Resources, Life Sciences, Marketing, Medical, Other, Technical Degree, Divorced, Married, and Single.

For our independent variable “Attrition”, ‘Yes’ was replaced with ‘1’ and ‘No’ was replaced with ‘0’. This was stored in “**y_variable”**.

Testing For Multicollinearity

Initially, we had run our models without checking for multicollinearity. However, after a first run of the models, it was clear that this was an important step which had to be reviewed prior to running these models. As well as an understanding that a common practice in regression analysis is to include only “n-1” of the dummy variables in the model to avoid multicollinearity.

A correlation matrix and variance inflation factor analysis are two common techniques to detect multicollinearity between the variables which were used during our analysis.

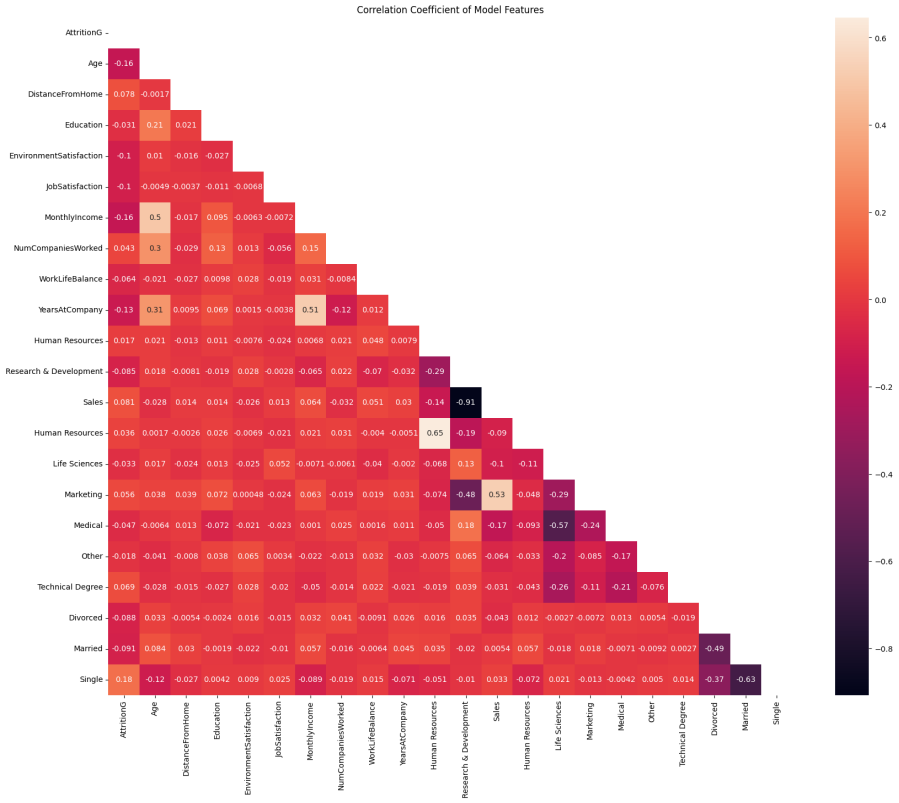
Variance Inflation Factor

```
# Examine the multicollinearity of the variables using Variance Inflation Factor
import statsmodels.stats.outliers_influence as oi

vif = pd.DataFrame()
vif["features"] = x_variables.columns
vif["VIF"] = [oi.variance_inflation_factor(x_variables.values, i) for i in range(x_variables.shape[1])]
print(vif)
```

```
                   features       VIF
0                       Age  1.524947
1          DistanceFromHome  1.006211
2                 Education  1.067055
3   EnvironmentSatisfaction  1.009610
4           JobSatisfaction  1.008527
5             MonthlyIncome  1.687043
6        NumCompaniesWorked  1.195671
7          WorkLifeBalance  1.014981
8           YearsAtCompany  1.470878
9           MaritalStatusG       inf
10         Human Resources       inf
11  Research & Development       inf
12                   Sales       inf
13         Human Resources       inf
14           Life Sciences       inf
15               Marketing       inf
16                 Medical       inf
17                   Other       inf
18        Technical Degree       inf
19                Divorced       inf
20                 Married       inf
21                  Single       inf
/usr/local/lib/python3.9/dist-packages/statsmodels/stats/outliers_influence.py:195: RuntimeWarning:

divide by zero encountered in double_scalars
```

A VIF of 1 indicates that there is no correlation between the variable and any of the other features present. A VIF > 5 or 10 indicates that the feature is likely contributing to multicollinearity. A VIF return of inf indicates that there is perfect correlation between two independent variables.

Correlation Matrix



Utilizing these two techniques our dataset was reduced to 15 features. The following was our final Variance Inflation Factor where we can see that no more features return “inf” or a VIF > 10.

```
# Examine the multicollinearity of the variables using Variance Inflation Factor
vif = pd.DataFrame()
vif["features"] = x_variables2.columns
vif["VIF"] = [oi.variance_inflation_factor(x_variables2.values, i) for i in range(x_variables2.shape[1])]
print(vif)
#We are good now!
```

	features	VIF
0	DistanceFromHome	2.203648
1	Education	7.405343
2	EnvironmentSatisfaction	5.863791
3	JobSatisfaction	5.582472
4	MonthlyIncome	4.197399
5	NumCompaniesWorked	2.354702
6	YearsAtCompany	3.302877
7	Human Resources	1.938262
8	Research & Development	3.925347
9	Human Resources	1.784082
10	Marketing	1.580478
11	Medical	1.714181
12	Other	1.141566
13	Technical Degree	1.197922
14	Divorced	1.465582
15	Single	1.664137

Iteration #1: Training & Testing Classification Models

Using the following code we split our data into a 70-30 where 70% of the data was used to train the models and 30% of the data was used to test the models.

```
# Split the dataset into training and testing sets
X3_train, X3_test, y3_train, y3_test = train_test_split(x_variables2,y_variable, test_size=0.3, random_state=42)
```

For this first iteration we chose “Logistic Regression”, “Decision Tree”, “Random Forest” and “ KNN” as these are common models used for binary independent variables. A “for” loop allowed us to fit all these models to our training set and validate using our testing set.

```
# Data accuracy rate of training data and test data in each classification algorithm.
for a,b in zip([lr2, dt2, rm2, knn2],["Logistic Regression","Decision Tree","Random Forest","KNN"]):
    a.fit(X3_train, y3_train)
    prediction= a.predict(X3_train)
    y3_pred    = a.predict(X3_test)
    score1    = accuracy_score(y3_train, prediction)
    score     = accuracy_score(y3_test, y3_pred)
    msg1      = "[%s] training data accuracy is : %f" % (b,score1)
    msg2="[%s] test     data accuracy is : %f" % (b,score)
    print(msg1)
    print(msg2)
```

```
[Logistic Regression] training data accuracy is : 0.843537
[Logistic Regression] test     data accuracy is : 0.866213
[Decision Tree] training data accuracy is : 1.000000
[Decision Tree] test     data accuracy is : 0.739229
[Random Forest] training data accuracy is : 1.000000
[Random Forest] test     data accuracy is : 0.850340
[KNN] training data accuracy is : 0.870748
[KNN] test     data accuracy is : 0.807256
```

Iteration #1: Results & Evaluation

The best way to compare our models is to use a combination of metrics that evaluate both the overall performance of the models and their ability to predict both classes. Relying solely on accuracy may not provide a complete picture of the model's performance. For instance, a model with high accuracy may have poor precision or recall, indicating that it is not performing well in correctly identifying one of the classes.

For example, in logistic regression, accuracy is calculated as the proportion of correctly predicted outcomes (true positives and true negatives) out of the total number of predictions. In the case of our data, if we had a model that would only predict “No”, our accuracy score would be 0.83. However, this model would not be very useful, as it is only predicting one class and is not learning from the data.

```
No     1233
Yes     237
Name: Attrition, dtype: int64
Active employees rate is  0.8388
Ex - employees   rate is  0.1612
```

Accuracy of Trained Models:

```
[Logistic Regression] training data accuracy is : 0.843537
[Logistic Regression] test     data accuracy is : 0.866213
[Decision Tree] training data accuracy is : 1.000000
[Decision Tree] test     data accuracy is : 0.752834
[Random Forest] training data accuracy is : 1.000000
[Random Forest] test     data accuracy is : 0.854875
[KNN] training data accuracy is : 0.870748
[KNN] test     data accuracy is : 0.807256
```

Precision of Trained Models:

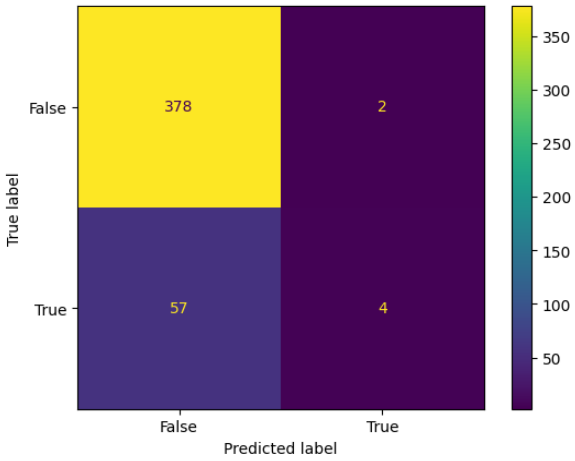
```
[Logistic Regression] training data precision is : 0.666667
[Logistic Regression] test precision is : 0.666667
[Decision Tree] training data precision is : 0.197674
[Decision Tree] test precision is : 0.197674
[Random Forest] training data precision is : 0.352941
[Random Forest] test precision is : 0.352941
[KNN] training data precision is : 0.200000
[KNN] test precision is : 0.200000
```

From these comparisons, the Logistic Regression model with C = 0.1 had a better performance when taking into account both accuracy and precision.

Confusion Matrix

The Confusion Matrix helps us understand the proportion of true positives and true negatives identified by the Logistics Regression model applied to the test dataset which contained 30% of the data from our original dataset. If we think about the goal of our model, we might want to tune this model to be able to

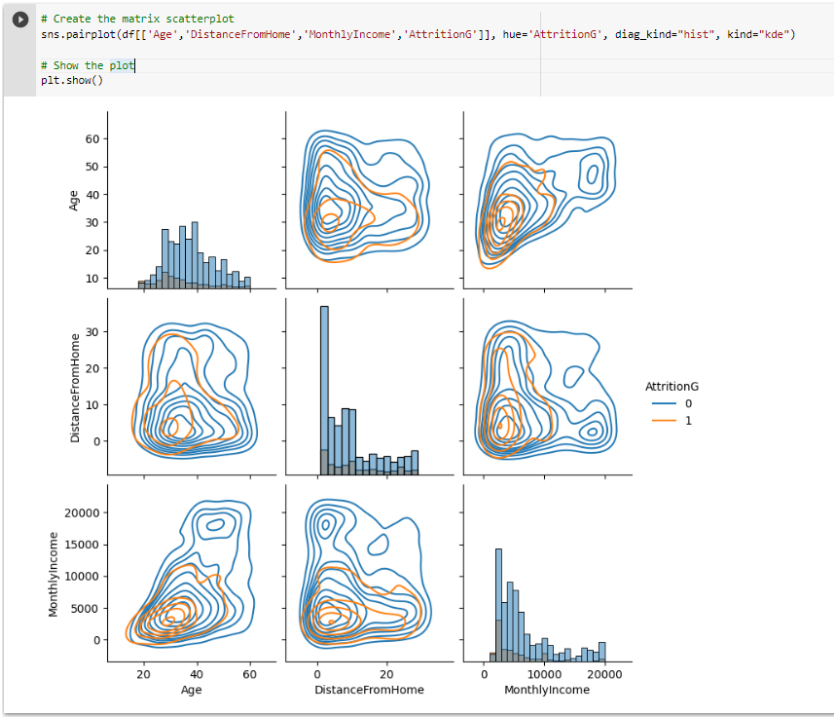
predict more True “Yes”, even if our False “Yes” increases, because when it comes to attrition, the cost of losing a person is higher than the cost of trying to retain a current employee. Hence, when selecting a model, context of the problem should also be taken into account along with other metrics showcased through this report.



Iteration #2: Training & Testing a Generalized Linear Model

Kernel Density Estimation (KDE) plots are useful for visualizing the distribution of a variable in a dataset. Unlike histograms, KDE plots can provide a smooth estimate of the density of a variable, which can help to identify patterns in the data that might be missed with discrete binning.

In this iteration, we are drawing KDE plots for four numeric variables (MonthlyIncome, DistanceFromHome, Attrition, and Age) and using the "Attrition" variable as the hue to distinguish between the Attrition categories. The kind parameter is set to "kde" to draw the KDE plots.



Next, we plot the Receiver Operating Characteristic (ROC) curve in a Generalized Linear Model (GLM) for predicting employee attrition. Plotting the ROC curve can help evaluate the model's ability to correctly classify employees who will leave the company (Attrition = “1”) and those who will not (Attrition = “0”). By looking at the ROC curve, we can see how well the model is able to balance between correctly identifying employees who will leave the company, and incorrectly identifying employees who will not leave for different decision thresholds.

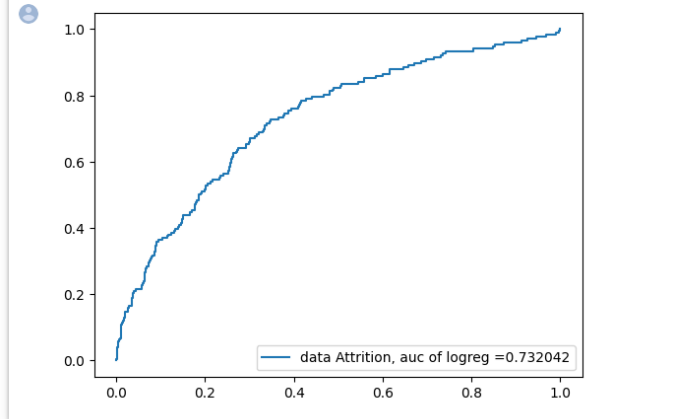
The following screenshot is of code used to plot the ROC curve for a logistic regression model that has been trained on the training set for the Employee Attrition dataset.

The “metrics.roc_curve()” function takes two arguments, “y_train” (actual target variable values of the training set) and “y_train_pred_final.Attrition_Prob” (predicted probability values of the target variable for the training set). It returns three values: FPR (false positive rate), TPR (true positive rate), and thresholds.

The “metrics.roc_auc_score()” function takes the same two arguments as “metrics.roc_curve()” and returns the Area Under the Curve (AUC) score for the ROC curve. The plt.plot() function is used to plot the ROC curve. It takes three arguments, FPR on the x-axis, TPR on the y-axis, and label (a string label for the plot). The plt.legend() function adds a legend to the plot, which is placed in the lower-right corner (loc=4). The plt.show() function displays the plot.

AUC can also be used as a metric for evaluating the model's performance, with an AUC of 0.5 indicating a random classifier and an AUC of 1.0 indicating a perfect classifier. In general, a higher AUC value indicates a better classifier. The higher the AUC score, the better the model. The plot shows how well the model is able to distinguish between the two classes by plotting the TPR against the FPR for different threshold values.

```
# plot the ROC curve in GLM model
fpr, tpr,  _ = metrics.roc_curve(y_train, y_train_pred_final.Attrition_Prob)
auc          = metrics.roc_auc_score(y_train, y_train_pred_final.Attrition_Prob)
plt.plot(fpr,tpr, label="data Attrition, auc of logreg ="+ str(round(auc,6)))
plt.legend(loc=4)
plt.show()
```



Iteration #2: Results & Accuracy

```
[ ] # The accuracy of data training in machine learning
    print(classification_report(y_train_pred_final.Attrition, y_train_pred_final.predicted))
```

```
                precision    recall  f1-score   support

             0       0.83      1.00      0.91       853
             1       0.75      0.03      0.07       176

      accuracy                           0.83      1029
     macro avg       0.79      0.52      0.49      1029
  weighted avg       0.82      0.83      0.76      1029
```

Above is a screenshot of the classification report for our binary classification model, where the goal is to predict whether an employee will churn (Yes = “1”) or not (No = “0”) based on our input features.

The precision for class “0” (employees who don't churn) is 0.83, meaning that out of all the instances predicted as not churning, 83% are actually true negatives. The recall for class “0” is 1.0, meaning that out of all the instances that are actually not churning, the model correctly identifies all of them as not churning. The F1-score for class “0” is 0.91, which is the harmonic mean of precision and recall, and is a measure of the overall performance of the model for this class.

For class “1” (employees who do churn), the precision is 0.75, meaning that out of all the instances predicted as churning, only 75% are actually true positives. The recall for class “1” is 0.03, meaning that out of all the instances that are actually churning, the model correctly identifies only 3% of them as churning. The F1-score for class “1” is 0.07.

The accuracy of the model is 0.83, meaning that it correctly predicts 83% of the instances in the test set. However, since the dataset is imbalanced with a much larger number of non-churning employees compared to churning employees, the accuracy can be misleading. Therefore, the macro and weighted averages of precision, recall, and F1-score are also reported, which take into account the performance of the model for both classes, weighted by the number of instances in each class.

In this case, the macro average F1-score is 0.49, indicating poor performance of the model. The weighted average F1-score is 0.76, which is slightly better, but still indicates that the model is not performing well for predicting churning employees.

References

1. Curic, A. (2022, November 30). *Employee turnover costly and a growing problem for Canadian businesses*. Employee Turnover Costly and a Growing Problem for Canadian Businesses. #CanadaEmployed. Retrieved April 3, 2023, from https://www.expresspros.com/CA/Newsroom/Canada-Employed/Employee-Turnover-Costly-and-a-Growing-Problem-for-Canadian-Businesses.aspx
2. Gopalani, P. (2022, December 19). Employee-Attrition-Rate: Employee Leaving organization why ? Create a Strategy to tackle this issue. Retrieved April 3, 2023, from https://www.kaggle.com/datasets/prachi13/employeeattritionrate