**Data Mining Assignment4 – Change of Primary Care Physician**

**Group No. 024**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Name (as appears in Canvas)** | **ID NO** | **Contribution (%)** |
| **1** | **IQBAL, MOHAMMED ZAHID** | **2020fc04118** | **100** |
| **2** | **KANTH, KOMMAJYOSULA VNS** | **2020fc04120** | **100** |
| **3** | **P, NARESHKUMAR** | **2020fc04122** | **100** |

**Problem Statement:**

An insurance provider (US based) offers health insurance to customers. The provider assigns a PCP(primary care physician) to each customer. The PCP addresses most health concerns of the customers assigned to them. For various reasons, customers want change of PCP. It involves significant effort for the provider whenever the customer makes a change of PCP.

You will find a subset of the insurance provider data along with PCP changes. The provider likes to understand why are members likely to leave the recommended provider. Further, they like to recommend a provider to them that they are less likely to leave.

**Solution needed:**

1. The reason for the attrition of the members leaving the recommended provider.
2. Suggest the Insurance provider to choose correct PCP for their members.
3. **Exploratory Data Analysis:**
   1. **Handling Missing Values:**

Claims\_day\_away attribute have a greater number of missing values. From the given dataset, we could see only 20% of records has data in the attribute.

Removing the attribute is not advisable since it has only few test records to predict. So, updated ‘0’ for all the missing values in claims\_day\_away attribute.

* 1. **Feature Reduction:**

Testindex column has unique values for all the records and it will not helpful for our model creation. So, Removing the Testindex feature from the dataset for the further proceedings.

* 1. **Understanding the datatypes:**

From the given dataset, we can see both categorical and numerical data.

|  |  |
| --- | --- |
| **Feature Name** | **Data Type** |
| Fqhc | Category – Nominal |
| Pcp\_lookback | Category – Nominal |
| Family\_assignment | Category – Nominal |
| Kid | Category – Nominal |
| Is\_ped | Category – Nominal |
| Same\_gender | Category – Nominal |
| Same\_language | Category – Nominal |
| Same\_address | Category – Nominal |
| Outcome | Category – Nominal |
| Tier | Category – Ordinal |
| Distance | Numerical - Continuous |
| Visit\_count | Numerical - Continuous |
| Claims\_Day\_away | Numerical - Continuous |
| Outcome | Category – Nominal (Class variable) |

* 1. **Data Analysis:**

Analyzing the numerical attributes based on the outcome variable. Taking mean of all the numerical attributes based on the outcome values to find the average.

data[["outcome","distance","visit\_count","Claims\_days\_away"]]

data.groupby('outcome').mean()

outcome distance visit\_count Claims\_days\_away

0 2.749683 1.032634 23.657676

1 5.266218 2.055118 33.070866

From the mean calculation and also after manually checks, we assume that we don’t have more outliers data in the attributes.

Plotting the graph for the categorical attributes to check each categorical value against the count of its outcome values.

get\_ipython().run\_line\_magic('matplotlib', 'inline')

pd.crosstab(data.tier,data.outcome).plot(kind='bar')

pd.crosstab(data.is\_ped,data.outcome).plot(kind='bar')

pd.crosstab(data.fqhc,data.outcome).plot(kind='bar')

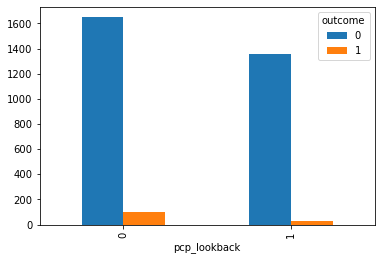
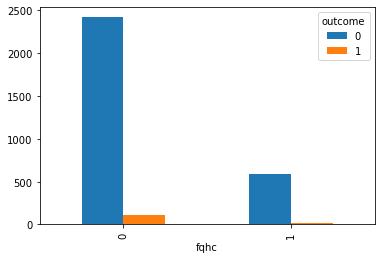
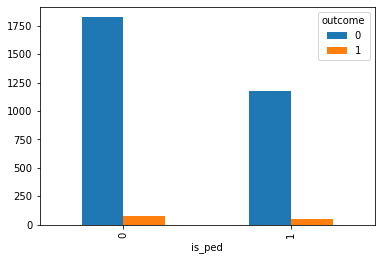
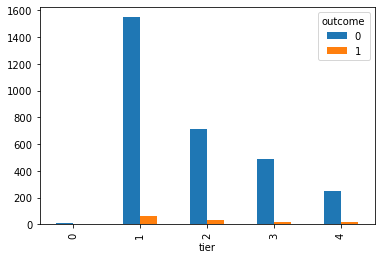
pd.crosstab(data.pcp\_lookback,data.outcome).plot(kind='bar')

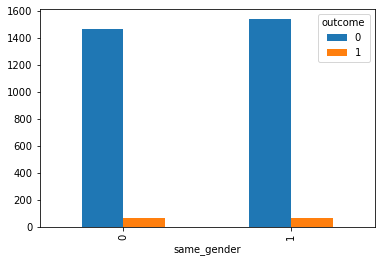
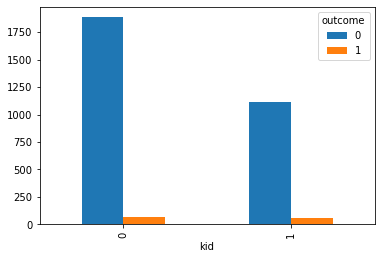
pd.crosstab(data.kid,data.outcome).plot(kind='bar')

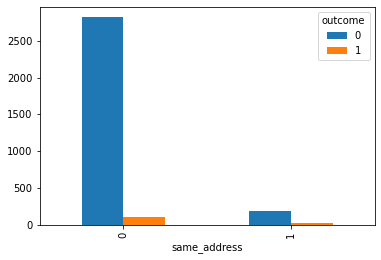
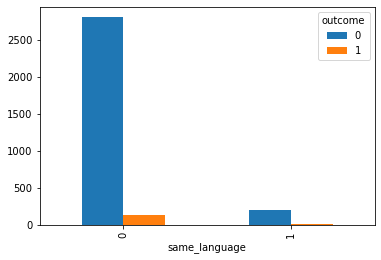
pd.crosstab(data.same\_gender,data.outcome).plot(kind='bar')

pd.crosstab(data.same\_language,data.outcome).plot(kind='bar')

pd.crosstab(data.same\_address,data.outcome).plot(kind='bar')







* 1. **Feature Selection and Feature Construction:**

Feature selection and new feature construction is not needed since the dataset has not enough data to choose the best feature. We will try to find the best feature after creating and testing the model.

1. **Building the Model:**

The preprocessing of given dataset is completed. Now, we are choosing the **Logistic regression algorithm** to create a new model since we have both categorical and numerical data in the dataset.

model = LogisticRegression(solver='liblinear', random\_state=0)

* 1. **Splitting Training data and Test data:**

In the given dataset, 80% of data has taken for training the model and 20% of data has taken for testing the model.

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=0)

**Train the data:**

80% of the dataset sends to newly created model to train it.

model.fit(x\_train,y\_train)

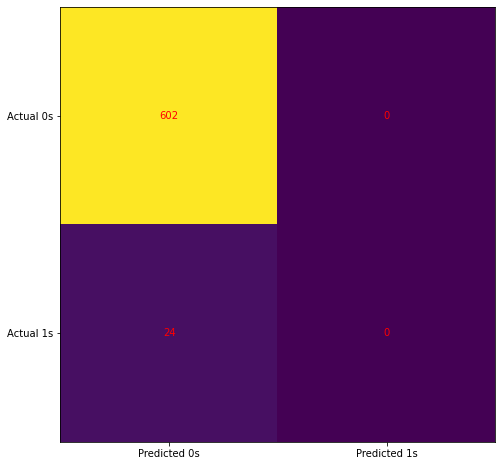
**Test the data:**

To test the model, sending the rest 20% of data to the model to predict the results.

y\_pred = model.predict(x\_test)

* 1. **Evaluating the model:**

**Confusion Matrix:**



**Classification Report:**

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.96 1.00 0.98 602

1 0.00 0.00 0.00 24

accuracy 0.96 626

macro avg 0.48 0.50 0.49 626

weighted avg 0.92 0.96 0.94 626

* 1. **Decision after evaluation:**

After testing the model, we could see the Precision as 96% for the outcome ‘0’ but 0% for the outcome ‘1’. The model predicting successful for the ‘0’ outcome and failing in the prediction for the ‘1’.

Due to low volume of outcome value ‘1’ in the dataset, the model is not predicting as expected. We need to find a way to balancing the imbalanced datasets.

1. **Next Iteration of Model Testing: (Refine the model)**
   1. **Handling Imbalanced datasets:**

Choosing RandomOverSampler strategy to increase the outcome value ‘1’ data to balance with outcome value ‘0’ data in the dataset. Due to this, the number of records in the dataset will be increased.

oversample = RandomOverSampler(sampling\_strategy=0.5)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=0)

os\_data\_X, os\_data\_y = oversample.fit\_resample(X\_train, y\_train)

* 1. **Calculate statistics of the test data:**

import statsmodels.api as sm

logit\_model=sm.Logit(y,X.astype(float))

result=logit\_model.fit()

print(result.summary2())

Current function value: 0.539718

Iterations 6

Results: Logit

==================================================================

Model: Logit Pseudo R-squared: 0.152

Dependent Variable: y AIC: 3422.2231

Date: 2021-03-10 19:21 BIC: 3488.8298

No. Observations: 3150 Log-Likelihood: -1700.1

Df Model: 10 LL-Null: -2005.0

Df Residuals: 3139 LLR p-value: 1.3876e-124

Converged: 1.0000 Scale: 1.0000

No. Iterations: 6.0000

------------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

------------------------------------------------------------------

fqhc -0.9075 0.1157 -7.8459 0.0000 -1.1342 -0.6808

pcp\_lookback -1.7559 0.1008 -17.4116 0.0000 -1.9536 -1.5583

family\_assignment 0.0461 0.0917 0.5022 0.6155 -0.1337 0.2259

kid 1.9812 0.2127 9.3146 0.0000 1.5643 2.3981

is\_ped -2.0615 0.2143 -9.6214 0.0000 -2.4815 -1.6416

same\_gender -0.4451 0.0755 -5.8972 0.0000 -0.5930 -0.2972

same\_language -0.5891 0.2170 -2.7153 0.0066 -1.0144 -0.1639

same\_address 1.6570 0.1371 12.0871 0.0000 1.3883 1.9256

distance 0.0284 0.0066 4.3071 0.0000 0.0155 0.0414

visit\_count 0.0674 0.0140 4.8259 0.0000 0.0400 0.0948

Claims\_days\_away 0.0012 0.0007 1.7621 0.0781 -0.0001 0.0025

==================================================================

* 1. **Feature Selection:**

From the above Logit table, the P value of family\_assignment, claims\_days\_away are greater than 0.05. So, we can exclude these features for the further processing.

* 1. **Training the model again:**

After balanced the dataset results and removing the dependent features, train the same logistic regression model using the 70% of dataset and test the predictions.

########### fitting the LR model

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

#########

ytrain\_pred = logreg.predict(X\_train)

y\_pred = logreg.predict(X\_test)

**Predicting the Model:**

######## Training confusion matrix

confusion\_matrix1 = confusion\_matrix(y\_train, ytrain\_pred)

print(confusion\_matrix1)

[[1366 111]

[442 286]]

**Testing the Model:**

######## Testing confusion matrix

from sklearn.metrics import confusion\_matrix

confusion\_matrix = confusion\_matrix(y\_test, y\_pred)

print(confusion\_matrix)

[[600 23]

[195 127]]

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.75 0.96 0.85 623

1 0.85 0.39 0.54 322

accuracy 0.77 945

macro avg 0.80 0.68 0.69 945

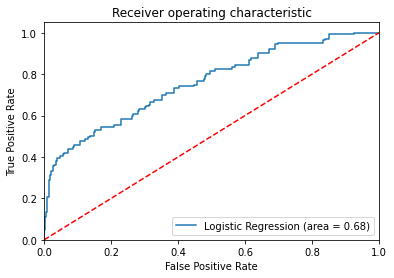
weighted avg 0.79 0.77 0.74 945

* 1. **Conclusion after testing the model:**

**Accuracy: 77%**

F1-Score: 85% for 0 and 54% for 1 (Score looks good for 0. But the score for 1 is reasonably less since we have less evidence of data in the dataset.)

ROC:



After balancing and selecting the right features, the model has enough data to predicting the data and the accuracy also quite good. So, no need to tune the model again and it is considered as final and correct model for this problem statement.

1. **Solution for this Problem Statement:**

------------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

------------------------------------------------------------------

fqhc -0.9075 0.1157 -7.8459 0.0000 -1.1342 -0.6808

pcp\_lookback -1.7559 0.1008 -17.4116 0.0000 -1.9536 -1.5583

kid 1.9812 0.2127 9.3146 0.0000 1.5643 2.3981

is\_ped -2.0615 0.2143 -9.6214 0.0000 -2.4815 -1.6416

same\_gender -0.4451 0.0755 -5.8972 0.0000 -0.5930 -0.2972

same\_language -0.5891 0.2170 -2.7153 0.0066 -1.0144 -0.1639

same\_address 1.6570 0.1371 12.0871 0.0000 1.3883 1.9256

distance 0.0284 0.0066 4.3071 0.0000 0.0155 0.0414

visit\_count 0.0674 0.0140 4.8259 0.0000 0.0400 0.0948

==================================================================

**Solution 1 - Reason for leaving the recommended provider:**

The following solutions made based on the co-efficient values of the above attributes. If the co-efficient value is positive then those attributes are the reason for the leaving the recommended provider.

1. The customers are more likely to leave if the member is a kid.
2. The customers are more likely to leave if the re-assigned provider has the same address as the provider pre-assigned.
3. The customers are more likely to leave if the distance between member and provider is longer.
4. The customers are more likely to leave if the visit count between member and provider is less.

**Solution 2 - Suggestion to reduce the attrition:**

1. Kids should be assigned to Pediatrician.
2. Don’t reassign the same provider to the customer.
3. Distance between member and provider should be less. So, the customer don’t need to travel long distance.
4. If the customer has less visit count with a physician then don’t reassign the same physician again.