DA-1

Abhishek H
abhishekhupele@gmail.com
8989935603
28 Sep, 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this Case is to Predict the Feedback. Let's assume it is Telecaller/Field Operative data for Customer Engagement for the purpose of

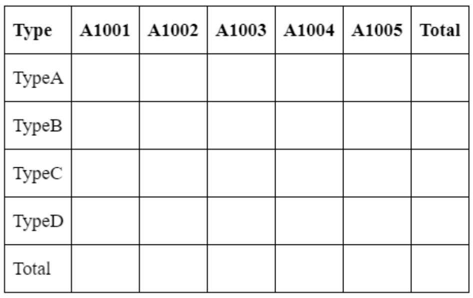
1. Maximising Sales/Revenue
2. Maximise customer touch point/increase brand awareness as a part marketing initiative/ hustle.

So, we are trying to get the FeedBack - Customer is Interested in purchase or sort of lead generation for

a) Conversion to Sales - if customer is positive to purchase
b) Pursue further for Conversion - if customer is undecided

We also do the below tasks as part of DA-1 :—

1. **<u>Create the summary table (refer attachment sample_table_DA1.png)</u>** using the sample dataset provided, programmatically. Clean the dataset if needed



Type	A1001	A1002	A1003	A1004	A1005	Total
TypeA						
TypeB						
TypeC						
TypeD						
Total						

2. **<u>Whether any inference could be reasoned out from the dataset.</u>**
3. <u>Use any machine learning algorithm(s) to predict the Feedback.</u> Reason out your choices.

1.2 Data

Data is an excel file with two sheet

SampleData1

	Id	Type	Date	CountA	CountB	CountC	FeedBack
1	10121	Email	8/4/16 10:16	0.01517016422	0.3859842	0.522609324665	Others
2	10122	Email	8/4/16 10:18	0.83029528784	0.1765000	0.071733055288	Others
3	10123	Email	8/4/16 10:30	0.40254570402	0.0519009	0.932725721587	Others
4	10124	Email	8/4/16 10:41	0.97441571076	0.2079939	0.350533017440	Others
5	10125	Email	8/4/16 10:59	0.02472098060	0.8079302	0.397741902549	Others

SampleData2		
	Id	User
1	10121	A1005
2	10122	A1002
3	10123	A1002
4	10124	A1002
5	10125	A1002

All columns details are below:

ID - LeadID

Type - Mode of Communication

Date - Time of Communication

CountA-C - Some arbitrary measurements for that Lead

User - User who makes the communication

FeedBack - Outcome of the communication (NC - No Contact, NI - Not Interested)

We need to combine both sheets on common column ‘Id’.

Now, We have total of 8 columns, in which we have 7 are independent variables and 1 dependent variable. ‘FeedBack’ (dependent variable) is the Outcome of communication (NC – No Contact, NI- Not Interested).

SampleData								
	Id	Type	Date	CountA	CountB	CountC	FeedBack	User
0	10121	Email	2016-08-04 10:16:47	0.773005	0.662162	0.622776	Others	A1005
1	10122	Email	2016-08-04 10:18:28	0.193826	0.870958	0.328081	Others	A1002
2	10123	Email	2016-08-04 10:30:24	0.772299	0.322429	0.456233	Others	A1002
3	10124	Email	2016-08-04 10:41:56	0.485769	0.128755	0.575060	Others	A1002
4	10125	Email	2016-08-04 10:59:00	0.015268	0.216612	0.098125	Others	A1002

Chapter 2

Methodology

2.1 Data Preprocessing: (Exploratory Data Analysis)

In Data Analytics, there is a famous quote by Riley Newman:

“More data beats better models. Better data beats more data.”

- Riley Newman

If we have our best model and we feed our data to that model, then it is not guaranteed that model will perform its best. As our data may have lots of noisy data (Garbage) and model will also follow noisy data and thus can produce wrong result because of that noisy data. We cannot remove noise/error completely from our data but we can reduce it with the help of EDA (Exploratory Data Analysis).

EDA involves getting summary of data with numerical statistics and Graphical visualization.

2.1.1 UnderstandingtheData First, we will look at our data and datatype:

We have 2 sheets ‘SampleData1’ (64 X 7) , ‘SampleData2’ (99 X 2) with common column as ‘Id’ . We will merge the 2 sheets to get ‘SampleData’ (66 X 8) and analyse the data.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 66 entries, 0 to 65
Data columns (total 8 columns):
Id         66 non-null int64
Type       64 non-null object
Date       66 non-null datetime64[ns]
CountA     66 non-null float64
CountB     66 non-null float64
CountC     66 non-null float64
FeedBack   64 non-null object
User       66 non-null object
dtypes: datetime64[ns](1), float64(3), int64(1), object(3)
memory usage: 4.6+ KB
```

** Python Code to generate above result

Time based variables: ‘Date’
Now what should be consider for them either continuous or categorical?

‘Date’ has date of 4th Aug 2018, so date is constant, but time of communication is different.

Categorical Variable:
‘Id’ is customer id and categorical. It is represented as datatype ‘int’ in SampleData, we will convert it to datatype ‘object’.
‘Type’ that is mode of communication by - Email, Visit, Call, to be updated, not available values
’User’ have 5 users values as 'A1001', 'A1002', 'A1003', 'A1004', 'A1005'

Continuous Variable:
‘CountA’, ‘CountB’, ‘CountC’ are continuous values. Some arbitrary measurements for that Lead.

Target variable:
‘FeedBack’ is our target variables and in categorical form. So our problem would be classification problem.

Let us now analyse our numerical data:

We will check summary of our numerical data or we say five point summary:

	CountA	**CountB**	**CountC**
count	66	66	66
mean	0.449231	0.487495	0.425085
std	0.28819	0.308335	0.270263
min	0.006013	0.001177	0.006691
25%	0.221218	0.21705	0.209038
50%	0.448362	0.544847	0.408826
75%	0.721173	0.723734	0.641617
max	0.981381	0.979509	0.976614

** Python Code to generate above result

Analysis:

We have four independent continuous variables.
As we can observe from above table, all independent variable are between 0.4 and 0.98.

<u>Checking categorical data Now</u>:

Counting of each unique value in each categorical variable.

```
value counts of categorical column

## Type ## :-
Update    37
Email     19
Call       7
Visit      1
Name: Type, dtype: int64
## User ## :-
A1003    22
A1001    18
A1002    15
A1004     8
A1005     3
Name: User, dtype: int64
## FeedBack ## :-
Others        29
NC            15
nc            13
UNDECIDED      2
Undecided      2
NI             1
undecided      1
undecided      1
Name: FeedBack, dtype: int64
```

[** Python Code to generate above result]

<u>Analysis</u>:

We have similar counting of unique value for 'Type', 'User', 'FeedBack'.
For 'Type' we have 37 (>50%) values as to be updated which can be considered as missing values. Further investigation to be done on the same for high missing values.
We should either find a way to impute the same accurately or drop the variable as >50% values are missing.

'User' A1005 (4%) and A1004(12%) contribute very less observations.

'FeedBack' data values need to be cleaned for multiple entries of same values (spelling error). We have values as 'Undecided' and 'Others'. For 'Undecided' can be taken up for follow-up communication but if we consider the data complete for the same day, it could not be kept in category : 'NI' customers for that day, it is safe to assume that category is Interested for this day and further data can be gathered to find out for the 'FeedBack' in follow-up calls.
'Others' category could be assumed as customer did not communicate feedback properly, or asked to call back, or decision maker was not available or reasons unknown. So, we will assume that effective contact could not be established with the client, thus as 'NC'

We find that ‘NI’ customer is only 1 observation, for sales perspective and this dataset let us club it with ‘NC’, not interested for sales - no contact .

Imbalance Class :
Also we have less observation for our ‘FeedBack’ interested class only 10% observations are related to this class. Generally, if minor class is less than 5% then there is serious issue of target imbalance. In our case we have 10% data but we may have issue of target imbalance. Will took this in consideration and will try to improve our model by removing target imbalance at the end.

2.1.1.1 Cleaning the Data

‘Type’ and ‘ FeedBack’ have maximum values for non-standard observation points. Let’s compare them :

Type	Call	Email	Update	Visit
FeedBack				
NC	NaN	NaN	15	NaN
NI	NaN	NaN	NaN	1
Others	7	19	1	NaN
UNDECIDED	NaN	NaN	2	NaN
Undecided	NaN	NaN	2	NaN
nc	NaN	NaN	13	NaN
undecided	NaN	NaN	1	NaN
undecided	NaN	NaN	1	NaN

** Python Code to generate above result

For ‘FeedBack’ as ‘Others’ , we see that mode of communication is ‘Email’, customer ‘NI’ is not likely to provide his/her feedback by mail. So, it is safe to assume as ‘NI’.

For ‘Type’ field as ‘Update’, we see that the ‘FeedBack’ is ‘NC’ for 28 observations , so likely it will be updated once ‘User’ has made communication to customer.

We clean ‘FeedBack’ datatype by rules :

Undecided -> Interested **‘IN’**
Others -> ‘NI’ -> ‘NC’ (only 1 observation in ‘NI’, we will club it to ‘NC’, as 19 abs no effective contact by Email)

We clean ‘Type’ mode of communication now by rules :

Update[‘NC’] -> ‘Email’ (as we are treating ‘Others’ with 19 observation ‘Email’ - no effective contact, so by same logic)
Update[‘Undecided’] -> ‘Call’ (as ‘I’ and more conversion by calling, assumption)

Update[‘Others’] -> ‘Call’ (assumption, for simplicity in coding, could be kept as ‘Visit’ on logical

	Id		
Type	Call	Email	Visit
FeedBack			
IN	6	NaN	NaN
NC	7	48	1

argument if found any)

** Python Code to generate above result

2.1.2 Missing value analysis

In our dataset let us check for any missing values. In case we have missing values then we should impute it using different method mean, median, KNN, linear regression, MICE, missForest etc.

Checking missing values for each column in our dataset:

```
Id 0
Type 2
Date 0
CountA 0
CountB 0
CountC 0
FeedBack 2
User 0
```

** Python Code to generate above result

Analysis:

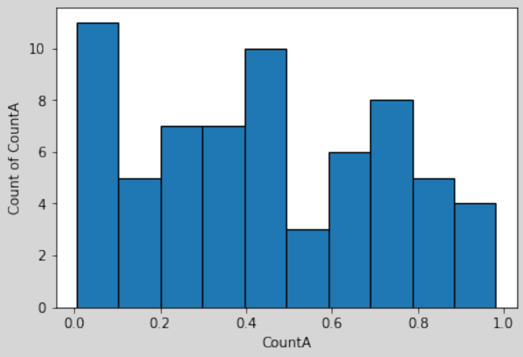
We will use fillna() method impute missing values in our dataset

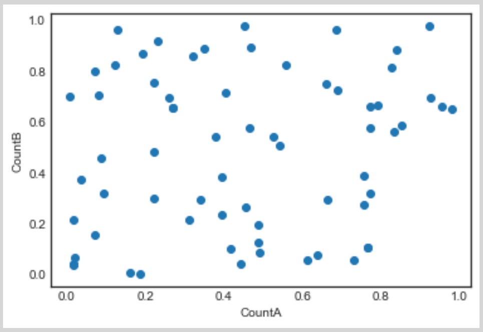
2.1.3 Visualising Data

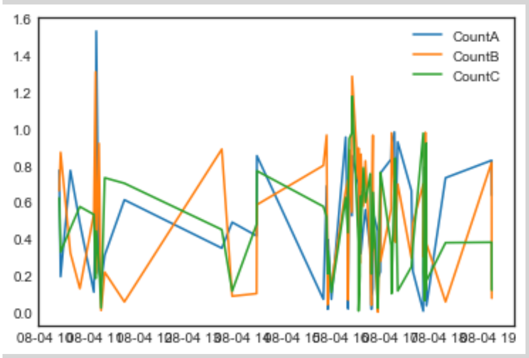
We visualise dataset by various graphs like scatter plot, box-plot, histogram, line charts to understand the relationships between data variables.

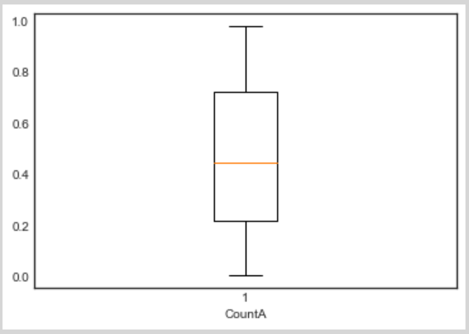
Below are some of the plots -

** Python Code to generate below result









2.1.4 Outlier Analysis:

Outlier detection and treatment is always a tricky part especially when our dataset is small. The box plot method detects outlier if any value is present greater than **(Q3 + (1.5 * IQR))** or less than **(Q1 – (1.5 * IQR))**

Q1 > 25% of data are less than or equal to this value

Q2 or Median -> 50% of data are less than or equal to this value

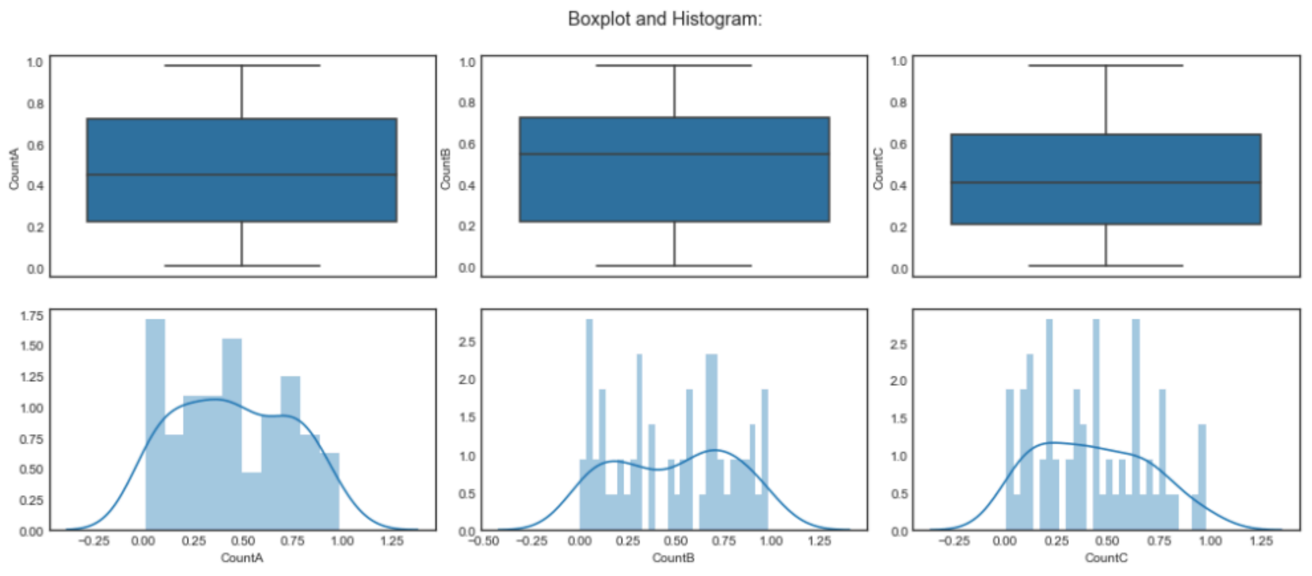
Q3 > 75% of data are less than or equal to this value

IQR(Inter Quartile Range) = Q3 – Q1

So, Boxplot method find approx. 1 % of data as outliers. It looks fine if we think only 1% of data we are treating as outlier and no impact would be after removal of outlier. Then we could be wrong.

Before treating outlier, we should look at nature of outlier. Is it information or outlier(error)? Let us check for outliers in our numerical dataset and will also look at distribution of data. So we have numerical columns as ‘CountA’, ‘CountB’, ‘CountC’.

Let us see box-plot and histogram for our numerical data.



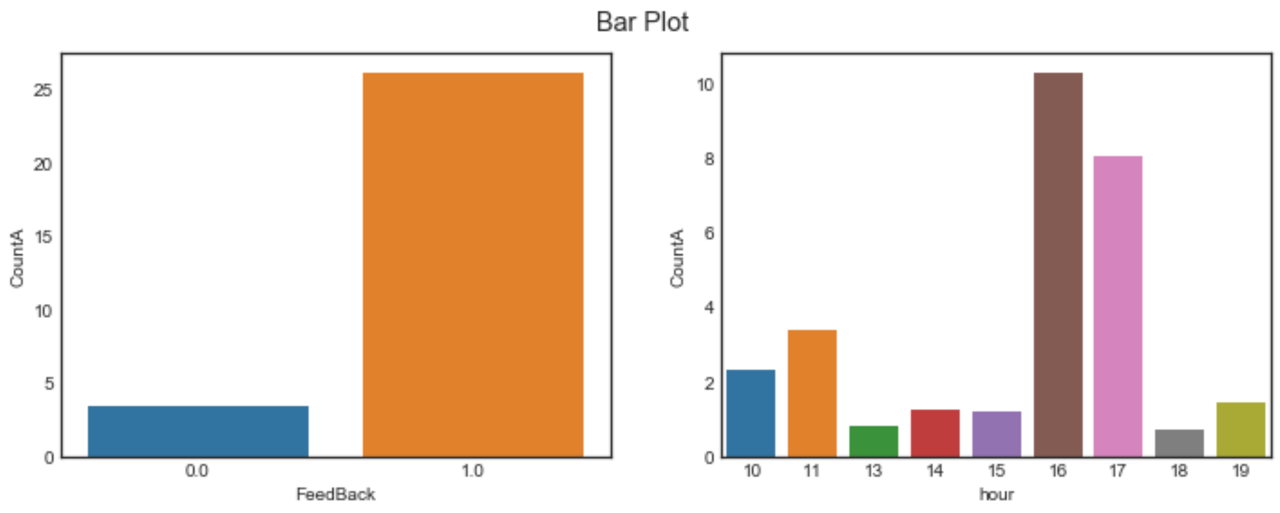
[** Python Code to generate above result](#)

Analysis:

We have no outliers .

2.1.5 Feature Engineering:

We have a column as ‘Date’, for time series data we can gain useful insights from it. The data is for Date : ‘ 4 Aug 2016’ which is constant but the time is variable. Time has 3 components hour, minute, seconds. For our analysis ‘hour’ of communication will be sufficient.



[** Python Code to generate above result]

Further , we will see that converting ‘hour’ datatype to categorical variable gives some significant pattern as in above graph. We will categorise the ‘hour’ into ‘time_of_day’ (morning, afternoon, night)

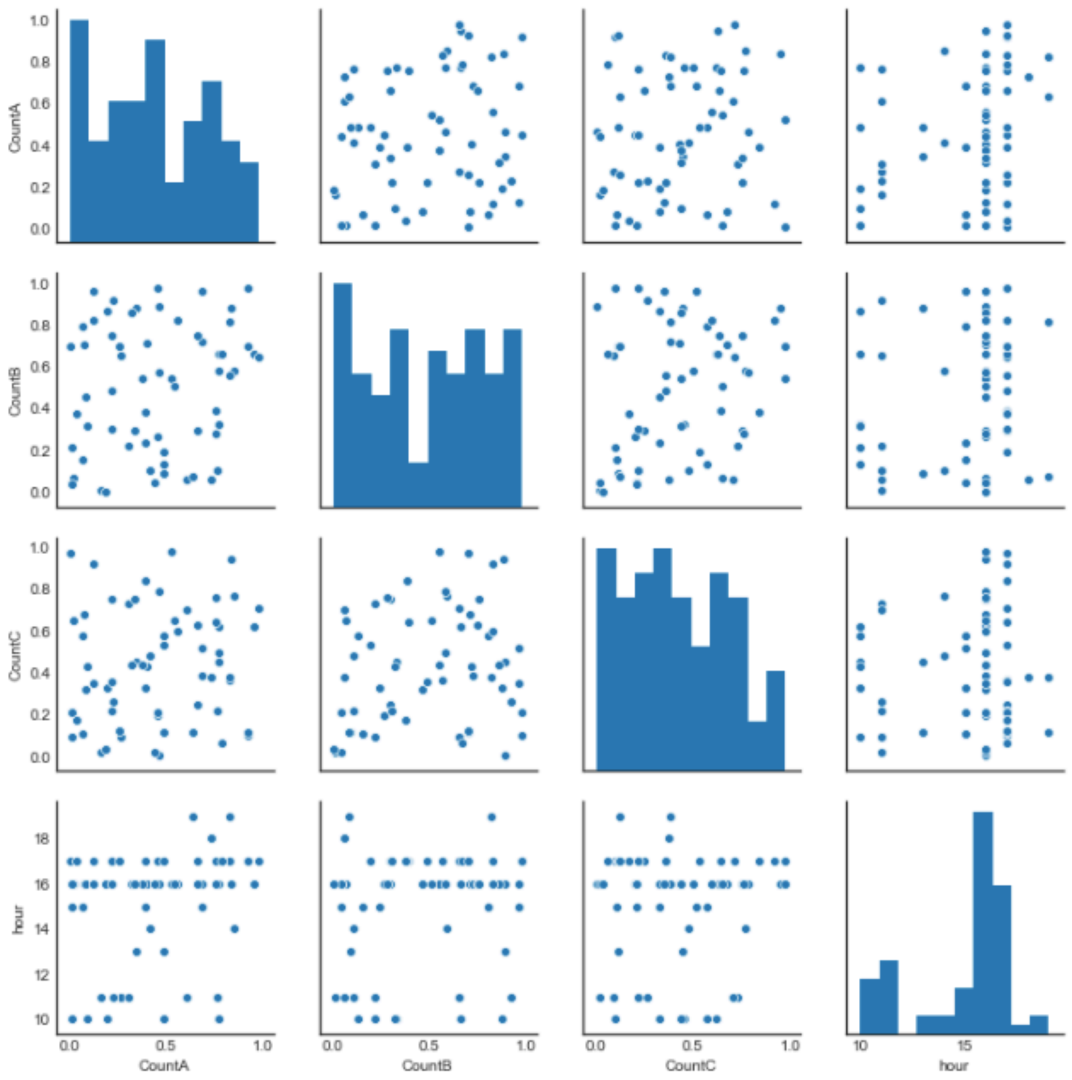
We will make dummy variables(with dropping one column) for our categorical independent variables .

2.1.6 Feature selection:

For feature selection of numerical variable we will see if there is multicollinearity between our continuous independent variable.

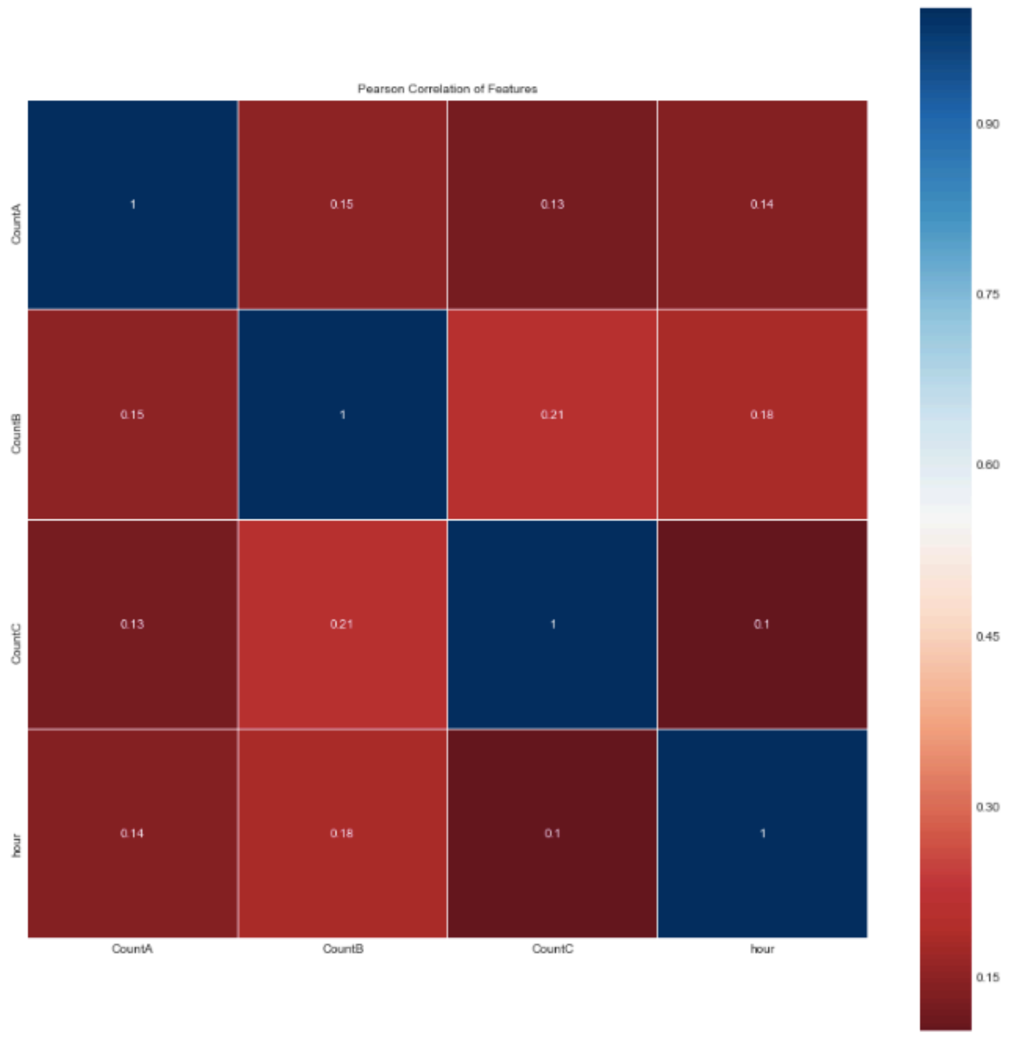
For this we will plot scatter plot between our continuous variables.

```
]: <seaborn.axisgrid.PairGrid at 0x10fd42d68>
```



** Python Code to generate above result

Let us see heatmap for the same:



[** Python Code to generate above result]

Analysis :

From heatmap and correlation plot we can see that , there is no multicollinearity present

Let us analyze for categorical variables also:

We have 3 categorical variable Type, User, time_of_day.

Let us first check these categorical variables that how much 'FeedBack' variable is dependent on other categorical variables. For this we would do chi-square (test of independence) test between 'FeedBack' column and these three columns:

Null Hypothesis for Chi-square test of Independence:
Two variables are independent.

Alternate Hypothesis:
Two variables are not independent.

```
Chi-square - test of independence
==================================
p-value between FeedBack and Type
2.0751633222057033e-05
----------------------------
p-value between FeedBack and User
0.9621643388283516
----------------------------
p-value between FeedBack and time_of_day
0.32487339451373193
----------------------------
```

** Python Code to generate above result

Analysis:

So, we want that our categorical variable should be independent. If we get p-value less than 0.05, it means we need to reject null hypothesis and accept alternate hypothesis which means variables are dependent. So, we would check for every combination and would print p-value.

From chi-sq test of independence, we have very less p-value than 0.05 for 'Type' with 'FeedBack' variable, so for both these case we have enough evidence to reject null hypothesis and accepting alternate hypothesis, that 'FeedBack' prediction is dependent on 'Type' or mode of communication.

But for 'User' and 'time_of_day' we have large p value saying that, 'FeedBack' and 'User' , 'time_of_day' are independent as we failed to reject null hypothesis. So we would drop 'User' and 'time_of_day' from our dataset.

Let us check whether 'Type' and 'User' and 'time_of_day' are independent or not?

For this also we use chi-sq test of independence test between them.

```
p-value between 'Type' and 'User'
0.2684637823731736
----------------------------
p-value between 'Type'  and 'time_of_day'
0.25441807217121043
----------------------------
p-value between 'User'  and 'time_of_day'
2.900514047022734e-05
----------------------------
```

** Python Code to generate above result

Analysis:

Now, we need to drop them to remove multicollinearity. But we have to be sure that we are not losing any information. So, we tried with dropping multicollinear column and building models and we got below result:

- on dropping weathersit or season we are losing accuracy with significant amount.
- On dropping holiday we are losing accuracy which is not significant amount.
- On dropping week_feat(weekday) or working day we are losing accuracy with little

amount. And if we drop holiday, that information is also included in working day.
We just can't be sure by looking at test result and deciding, we need to get all factors. Here two columns most of the time showing collinearity but some datapoints would not be showing collinearity and at that datapoint there could be abrupt difference in bike renting count column which is very important information for our model.
We will drop only holiday column as we have working day column which has information of holiday also, that is why on dropping holiday we are not losing our accuracy. Also in while getting important features we are getting last ranking for holiday column with very less importance.

Let us check now important feature of our dataset:

	Feature	importance
0	Type	0.423475
1	CountA	0.178135
2	CountC	0.138901
3	CountB	0.13743
4	hour	0.12206

** Python Code to generate above result

Let us check again VIF value after removal of mulitcollinear columns :

```
const     37.9
CountA      1.0
CountB      1.1
CountC      1.1
hour       1.1
dtype: float64
```

** Python Code to generate above result

Analysis :

VIF should be less than 10 for multicollinearity. Const is not part of our dataset, it is added for calculation of VIF. So as per VIF values, now we don't have multicollinearity in our dataset.

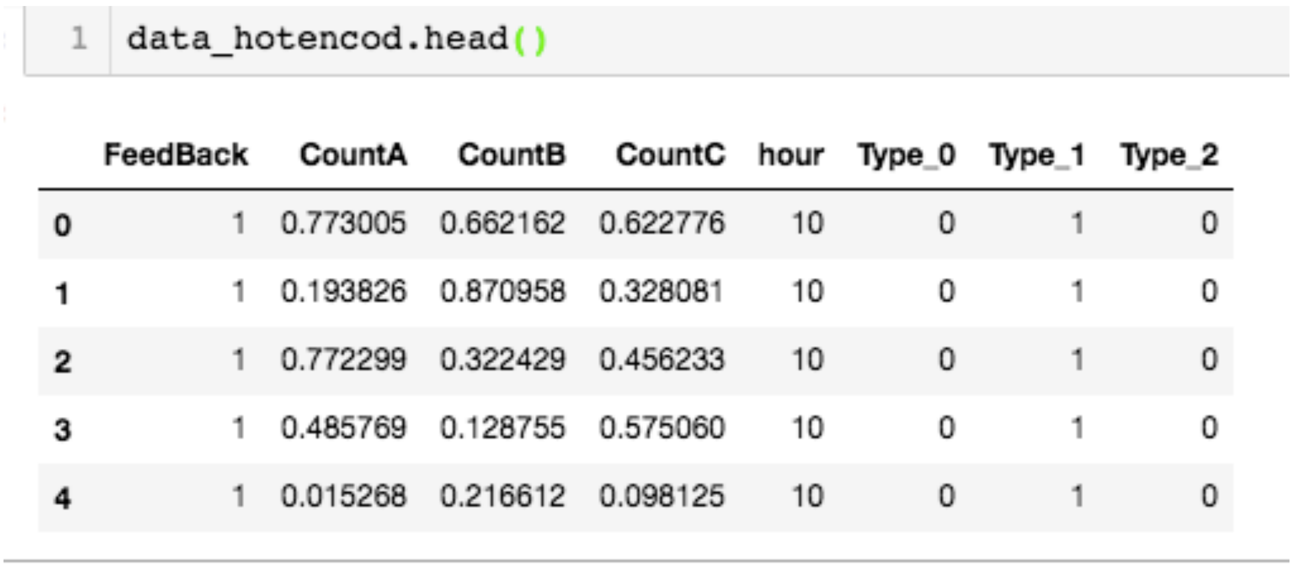
Now, we have good value of vif and don't have multicollinearity. Const is not part of our dataset it was just added as it is required for calculation of VIF.

2.1.7 Feature Scaling:

Our dataset has all values in almost same range, so we don't require feature scaling for this dataset. Feature scaling is important for those variables for which their values are too high and too low. Algorithm which uses distance method, are affected by out of range values. Higher value dominates the lesser values in calculating distance. But in our case min value is 0.4 and maximum is 0.98.
So, Feature scaling is not requiring for this case.

2.1.8 Data After EDA

We would remove 'Id', 'User', 'Date', 'time_of_day', and add 'hour' variable. We will make dummy variables for 'Type' in python.



```
data_hotencod.head()
```

	FeedBack	CountA	CountB	CountC	hour	Type_0	Type_1	Type_2
0	1	0.773005	0.662162	0.622776	10	0	1	0
1	1	0.193826	0.870958	0.328081	10	0	1	0
2	1	0.772299	0.322429	0.456233	10	0	1	0
3	1	0.485769	0.128755	0.575060	10	0	1	0
4	1	0.015268	0.216612	0.098125	10	0	1	0

2.2 Modeling

We will now build our models. Before proceeding please look at below key terms to avoid any confusion in next steps.

➢ data_hotencod : training dataset containing all observations.
➢ X_train : containing independent variables of data_hotencod
➢ y_train: containing dependent variable (FeedBack) of data_hotencod
➢ X_resampled : dataset after applying SMOTE + Tomek oversampling process on data_hotencod It is only containing independent variables.
➢ y_resampled : dataset after applying SMOTE + Tomek oversampling process on data_hotencod. It is only containing dependent variable (FeedBack).

2.2.1 K-fold CV and GridsearchCV

Before building models on our dataset, we would like to explore two things:

- K-fold cross validation
- GridSearchCV

K-fold Cross Validation:

K-fold cross validation is used to check performance of model which is checked on K different test dataset. Let us assume, we have built a model and we are checking performance of our model on a test data and our model show accuracy of 95% and now we will check our model on different test data and now accuracy is 80%. So what should we consider for deciding model performance? So in this, K-fold cross validation helps, it would divide our training data in k sets and will build a model using k-1 training set and one left set would be used to test our model performance. In this way it would build k times model and each time there would be different test dataset to check performance and at the end all k model's accuracy mean value would be considered as model accuracy.
So, we would use K-fold cross validation technique to get performance of our model.

GridsearchCV : (Hyperparameter tuning)

Hyperparameter are the parameters which we pass as argument to our building function, like kernel, criterion, n_estimators etc. So to get best values of these gridserchcv is used. In this technique, we make list of these different parameters and then gridsearchcv build model for every combination of these parameters and then check crossvalidation score and based on score it gives the best combination of hyperparameters.

And then we can build our model with the values of hyperparameter given by GridSearchCV. This is called performance tuning and we would use this to tune our model.

2.2.2 Building models

Models and performance of models :

We will build one by one all models and will check performance of our model and then at the will decide for which model we should go.

Logistic Regression:

Performance of Logistic Regression model (K-fold CV and score on test dataset)

```
K-fold cross validation score of model for k = 10 is :
0.9166666666666667
====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.00      0.00      0.00         1
          1       0.93      1.00      0.96        13

avg / total       0.86      0.93      0.89        14

====== Confusion matrix =======
[[ 0  1]
 [ 0 13]]
```

** Python Code to generate above result

Analysis:

We got performance of 91% for dataset i.e. 91% K- fold accuracy. Logistic regression gives better result for linearly separable data. Let us try other model also and at the end we would decide our model based on our performance. 0 observations are predicted as Interested and in actual 1 observation was Interested but predicted as opposite/True.

KNN(K – Nearest Neighbors) :

```
Performance of KNN model (K-fold CV and score on test dataset)

K-fold cross validation score of model for k = 10 is :
0.9016666666666667
======================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.00      0.00      0.00         1
          1       0.92      0.85      0.88        13

avg / total       0.85      0.79      0.82        14
====== Confusion matrix =======
[[ 0  1]
 [ 2 11]]
```

** Python code to generate above result

Analysis:
We got 90% performance for dataset i.e. % K-fold accuracy and slightly less accurate than logistic regression. 2 observations are predicted as FeedBack False and in actual these observations having FeedBack as True.

Naïve Bayes :

Performance of Naïve Bayes model (K-fold CV and score on test dataset)

```
K-fold cross validation score of model for k = 10 is :
0.9183333333333333
======================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.20      1.00      0.33         1
          1       1.00      0.69      0.82        13

avg / total       0.94      0.71      0.78        14

====== Confusion matrix =======
[[1 0]
 [4 9]]
```

** Python code to generate above result

Analysis:
We got 92% performance for dataset i.e. K-fold accuracy and slightly more accurate than logistic regression. 1 observations are predicted as False and in actual these observations having FeedBack as False.

Decision tree:

Performance of Decision Tree model (K-fold CV and score on test dataset

```
K-fold cross validation score of model for k = 10 is :
0.9216666666666666
=====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.33      1.00      0.50         1
          1       1.00      0.85      0.92        13

avg / total       0.95      0.86      0.89        14

====== Confusion matrix =======
[[ 1  0]
 [ 2 11]]
```

** Python code to generate above result

Analysis:
We got 92% K-fold accuracy for dataset . We got slightly higher performance for dataset with outliers than dataset without outliers. 2 observations are predicted as FeedBack False and in actual these observations having Feedback as True.

Random forest:

Performance of Random Forest model (K-fold CV and score on test dataset)

```
K-fold cross validation score of model for k = 10 is :
0.925
=====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.33      1.00      0.50         1
          1       1.00      0.85      0.92        13

avg / total       0.95      0.86      0.89        14

====== Confusion matrix =======
[[ 1  0]
 [ 2 11]]
```

** Python code to generate above result

Analysis:
We got slightly high performance for dataset i.e. 92.5% K-fold accuracy.

2.2.3 Hyperparameter tuning

Hyperparameter tuning is used to find optimum values of arguments used in building models like n_estimators, max_depth, kernel etc. so that we could gain better result with these tuned parameter. So we will do hyperparameter tuning for two models which gave us accuracy more than 90% i.e. Decision Tree Classifier and Random Forest.

Decision Tree Model Hyperparameter tuning:

Let us tune decision tree for following parameters on dataset

```
# hyperparameter tuning for Decision tree classifier
from sklearn.model_selection import GridSearchCV
FeedBack_classifier = DecisionTreeClassifier(random_state=1)
params = [{'criterion':['entropy', 'gini'],
        'max_depth':[6,8,10,12,20],'class_weight':['balanced',{0:0.45, 1:0.55},
                {0:0.55,1:0.45},{0:0.40,1:0.60}],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=FeedBack_classifier, param_grid=params,
                    scoring = 'f1', cv = 10, n_jobs=-1)

# tuning Decision Tree for dataset
grid_search = grid_search.fit(X_train, y_train)
grid_search.best_params_
```

```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_depth': 6,
 'random_state': 1}
```

** Python code to generate above result

Now, building DecisionTreeclassifier with parameter suggested by GridSearchCV for dataset

```
K-fold cross validation score of model for k = 10 is :
0.9383333333333332
=====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.00      0.00      0.00         1
          1       0.90      0.69      0.78        13

avg / total       0.84      0.64      0.73        14

====== Confusion matrix =======
[[0 1]
 [4 9]]
```

** Python code to generate above result

<u>Random Forest Model Hyperparameter tuning</u>:

Let us tune Random Forest for following parameters on dataset

```
############ Hyperparameter tuning for Random Forest #############
# Grid search for finding best parameter for random_forest on FeedBack_data_df dataset
FeedBack_classifier = RandomForestClassifier(random_state=1)
params=[{'criterion':['entropy', 'gini'],'n_estimators':[800,1000],
        'max_depth': [8, 10, 12], 'class_weight':['balanced', {0:0.45, 1:0.55},
                   {0:0.55, 1:0.45}],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=FeedBack_classifier, param_grid=params,
                       scoring = 'f1', cv = 10, n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
grid_search.best_params_
```

```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_depth': 8,
 'n_estimators': 800,
 'random_state': 1}
```

[** Python code to generate above result]

Let us now build again Random Forest model with parameter suggested by GridsearchCV

```
K-fold cross validation score of model for k = 10 is :
0.925
=====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       0.20      1.00      0.33         1
          1       1.00      0.69      0.82        13

avg / total       0.94      0.71      0.78        14

====== Confusion matrix =======
[[1 0]
 [4 9]]
```

[** Python code to generate above result]

2.2.4 SMOTE + Tomek (Oversampling)

As we see, we improved our model with tuning of hyperparameter. Overall accuracy increased but we can see from final confusion matrix we have more false positive than false negative. (assuming positive class to FeedBack false i.e. 0)

False positive:- Incorrect classified as class 0 (FeedBacking false) , in actual they belongs to class 1 (FeedBacking True).

So, from business point of view, we would be more interested in person who may FeedBack. So that, we can give extra attention to those customers for retaining them with our business.

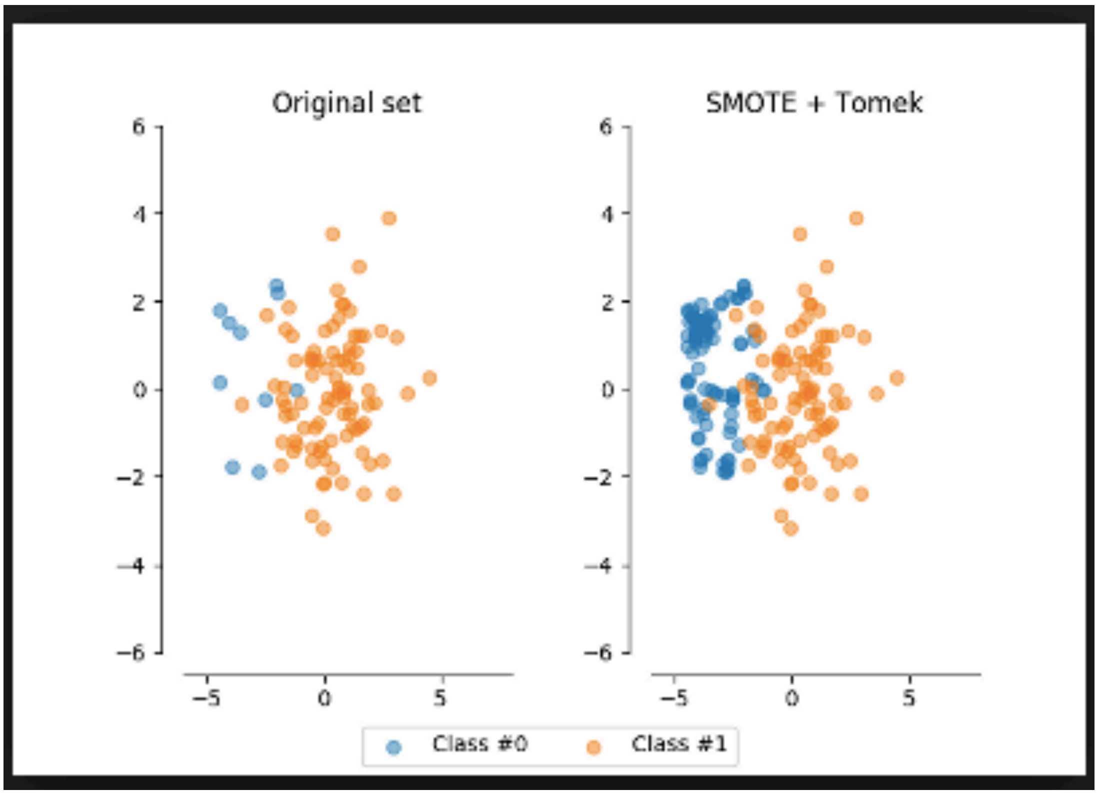
As, for class 0 i.e FeedBacking false, we are getting more accuracy than class 1 i.e. FeedBacking true. And we have only 14% data for FeedBacking true class. So, it seems our model is overfitting to class 0 i.e. FeedBacking false.

To resolve this issue let us make target in balanced way. For this we would use oversampling technique of SMOTE (Synthetic minority over-sampling technique) and to remove noisy over data. Both SMOTE + tomek will make oversampled data without noise. SMOTE generate artificial data for minority class, for each observation for minority class k nearest neighbors are identified and then randomly few neighbors are selected and then artificial observation are generated and spread along the line joining observation and nearest neighbors.

Tomek (T–link) use distance method between points and based on distance between good examples identifies each observation as data, noise and at boundary points. With the help of Tomek we can remove noise.

We will use SMOTE + Tomek algorithm for balancing target class.

Below picture is from sklearn official documentation for SMOTE + Tomek



- We will implement smote + tomek in dataset
- We will use Random Forest only and will tune its hyperparameter.

Shape of dataset after SMOTE + Tomek (oversampling)

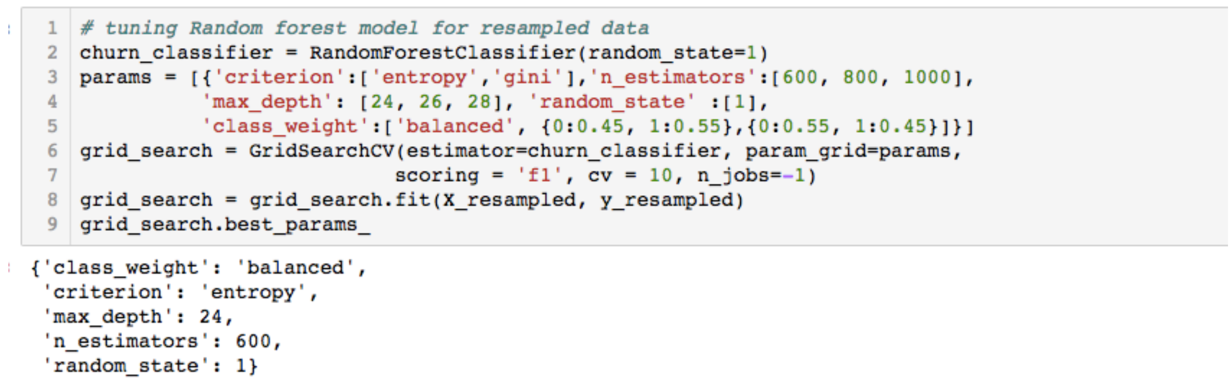
```python
# checking shape of data after resampling
print(X_resampled.shape)
print(y_resampled.shape)
print("class proportion")
print(pd.Series(y_resampled).value_counts(normalize = True))
```

```
(120, 7)
(120,)
class proportion
1     0.5
0     0.5
dtype: float64
```

[**Python code for resampling using smotetomek](#)

Random Forest Hyperparameter tuning on oversampled dataset

Now, Tuning Random Forest Model for first resampled Data



```python
# tuning Random forest model for resampled data
churn_classifier = RandomForestClassifier(random_state=1)
params = [{'criterion':['entropy','gini'],'n_estimators':[600, 800, 1000],
           'max_depth': [24, 26, 28], 'random_state' :[1],
           'class_weight':['balanced', {0:0.45, 1:0.55},{0:0.55, 1:0.45}]}]
grid_search = GridSearchCV(estimator=churn_classifier, param_grid=params,
                           scoring = 'f1', cv = 10, n_jobs=-1)
grid_search = grid_search.fit(X_resampled, y_resampled)
grid_search.best_params_
```

```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_depth': 24,
 'n_estimators': 600,
 'random_state': 1}
```

** Python code to generate above result

Building Random Forest with parameter suggested by GridsearchCV for oversampled dataset

```
K-fold cross validation score of model for k = 10 is :
0.9666666666666666
=====================================
====== Classification Report =======
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         1
          1       1.00      1.00      1.00        13

avg / total       1.00      1.00      1.00        14

====== Confusion matrix =======
[[ 1  0]
 [ 0 13]]
```

** Python code to generate above result

Analysis:
After oversampling data, we improved our model and moreover now our model is predicting true FeedBack more than previous model and dataset.
We got 96.67% K-fold accuracy for dataset

Final accuracy on test dataset:

	FeedBack_prediction	
	0	1
Actual class 0:	1	0
Actual class 1:	0	13

Accuracy = Correct Prediction / total observation
= (1+13)/(1+0+0+13)
= 100%

Model K-fold accuracy for K = 10: 96.67%

Chapter 3

Conclusion

3.1 Create the summary table (DA1.png):

	CountA	CountB	CountC
User			
A1001	0.514366	0.632827	0.405684
A1002	0.396569	0.332355	0.37591
A1003	0.465083	0.55416	0.485652
A1004	0.309543	0.250032	0.331924
A1005	0.577985	0.535551	0.591643

** Python code to generate above result

3.2 Inference reasoned out from the dataset

Dataset:
- First take two sheets and merge them to single dataset.
- Columns 'FeedBack' and 'Type' contain incomplete information, imputed with some assumptions
- Drop columns 'Id', 'Date', 'User'
- Change 'Type', 'Feedback' columns to category and then to levels of category
- Created new variable hour
- Imbalanced dataset : Apply SMOTE + Tomek to balancing the target variable on training dataset.

Pls refer below link for detailed analysis :

2.1 Data Preprocessing: (Exploratory Data Analysis)

3.3 Final Model and Training Dataset

Model:
- Use prepared dataset to create balanced dataset
- Use random Forest model and train using dataset which we prepared with above steps.
- Do hyperparameter tuning.
- And then build model using tuned hyperparameter.
- Our model is ready to predict !!!!!!!!!

Pls refer below link for detailed modelling steps :

2.2 Modeling

3.4 End Notes

Result shown in this report are from Python notebook.
If we would have large dataset then we would get more dependable outcome.

Proficient in doing it in R-code also. Result from R code would not be exact same for building models as implementation of function at backend is different for R and Python. But information would be almost same for both Python and R.

Complete Python code

```
# Project for Classification of FeedBack based on Communication Data
# importing libraries
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns


# reading Excel file
SampleData1 = pd.read_excel('DA-1.xlsx', 'SampleData1',header=0)
SampleData2 = pd.read_excel('DA-1.xlsx', 'SampleData2',header=0)

# Merging 2 Sheets
SampleData = pd.merge(SampleData1,SampleData2)

##########################################
#                                        #
#     2.1 Exploratory Data Analysis      #
#                                        #
##########################################

##################################
#  2.1.1 understanding the data  #
##################################

#Checking few observation of dataset
SampleData.head()

# looking at information of dataset ->  see output
SampleData.info()

SampleData['Id'] = SampleData['Id'].astype('O')
SampleData1.shape, SampleData2.shape, SampleData.shape

# looking at five point summary for our numerical variables ->  see output
SampleData.describe()

# counting of each unique values in each categorical variable -> see output
print("value counts of categorical column")
print()
for i in ['Type', 'User', 'FeedBack'] :
    print('## {} ## :- '.format(i))
    print(SampleData[i].value_counts())

# Comparing observations for 'FeedBack' and 'Type' -> see output
SampleData.pivot_table(index='FeedBack', columns='Type',
                aggfunc={'Id':'count'})
```

```
###################################
#  2.1.2 Missing value analysis   #
###################################

# checking for missing values in dataset -> see output
for i in SampleData.columns :
    print(i, SampleData[i].isnull().sum())


# filling NaN values in FeedBack
t = pd.Series(SampleData['FeedBack'])
SampleData['FeedBack'] = t.fillna('NC')
SampleData.tail()

# Cleaning the Data
for i in range(SampleData.shape[0]) :
    if SampleData.ix[i,'FeedBack'] == 'UNDECIDED':
        SampleData.ix[i,'FeedBack'] = 'IN'
    elif SampleData.ix[i,'FeedBack'] == 'undecided':
        SampleData.ix[i,'FeedBack'] = 'IN'
    elif SampleData.ix[i,'FeedBack'] == 'undecided ':
        SampleData.ix[i,'FeedBack'] = 'IN'
    elif SampleData.ix[i,'FeedBack'] == 'Undecided':
        SampleData.ix[i,'FeedBack'] = 'IN'
    elif SampleData.ix[i,'FeedBack'] == 'nc':
        SampleData.ix[i,'FeedBack'] = 'NC'
    elif SampleData.ix[i,'FeedBack'] == 'Others':
        SampleData.ix[i,'FeedBack'] = 'NC'
    elif SampleData.ix[i,'FeedBack'] == 'NI':
        SampleData.ix[i,'FeedBack'] = 'NC'
SampleData['FeedBack'].value_counts()
#SampleData.pivot_table(index='FeedBack', columns='Type', aggfunc={'Type':'count'})

# Cleaning the Type Data
for i in range(SampleData.shape[0]) :
    if SampleData.ix[i,'Type'] == 'Update':
        if SampleData.ix[i,'FeedBack'] == 'NC':
            SampleData.ix[i,'Type'] = 'Email'
    if SampleData.ix[i,'Type'] == 'Update':
        if SampleData.ix[i,'FeedBack'] == 'IN':
            SampleData.ix[i,'Type'] = 'Call'

# filling NaN value in Type variable
l = pd.Series(SampleData['Type'])
SampleData['Type'] = l.fillna(method = 'ffill')
SampleData.tail()




###################################
### 2.1.3 Visualising Data                         #
################################### -> see output
```

```
###########################################
#  3.1 Create the summary table (DA1.png):            #      ->   see output
###########################################

SampleData.groupby(‘User')[['CountA','CountB','CountC']].mean()

# Histogram
plt.figure()
plt.hist(SampleData['CountA'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of CountA')
plt.xlabel('CountA')

plt.figure()
plt.hist(SampleData['CountB'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of CountB')
plt.xlabel('CountB')

plt.figure()
plt.hist(SampleData['CountC'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of CountC')
plt.xlabel(‘CountC')

# Histogram
plt.figure()
plt.hist(SampleData['User'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of User')
plt.xlabel('User')

plt.figure()
plt.hist(SampleData['Type'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of Type')
plt.xlabel('Type')

plt.figure()
plt.hist(SampleData['FeedBack'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of FeedBack')
plt.xlabel(‘FeedBack')


# Scatter Plot
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-white')

plt.figure()
plt.plot(SampleData['CountA'],SampleData['CountB'],'o')
plt.xlabel('CountA')
plt.ylabel('CountB')

plt.figure()
plt.plot(SampleData['CountB'],SampleData['CountC'],'o')
plt.xlabel('CountB')
plt.ylabel('CountC')
plt.figure()
plt.plot(SampleData['CountA'],SampleData['CountC'],'o')
plt.xlabel('CountA')
plt.ylabel('CountC')

plt.figure()
plt.plot(SampleData['User'],SampleData['CountB'],'o')
plt.xlabel('User')
plt.ylabel(‘CountB')

# Scatter Plot
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-white')

plt.figure()
plt.plot(SampleData['CountA'],SampleData['CountB'],'o')
plt.xlabel('CountA')
plt.ylabel(‘CountB')

# Line Chart
d = SampleData.pivot_table(index='Date',aggfunc={'CountA':sum})
d = d.reset_index()
plt.plot(d['Date'],d['CountA'])
d = SampleData.pivot_table(index='Date',aggfunc={'CountB':sum})
d = d.reset_index()
plt.plot(d['Date'],d['CountB'])
d = SampleData.pivot_table(index='Date',aggfunc={'CountC':sum})
d = d.reset_index()
plt.plot(d['Date'],d['CountC'])
plt.legend()

# Boxplot for Outlier Analysis
num_var = ['CountA', 'CountB', 'CountC']
for i in num_var:
    plt.figure()
    plt.boxplot(SampleData[i])
    plt.xlabel(i)






# Dropping Id and Date Variable
# Date Variable will be used in Feature Engineering
temp1 = SampleData.copy()
SampleData = SampleData.drop([‘Date','Id'],axis=1)

# Converting Categorical to levels
from fancyimpute import KNN
temp = SampleData.copy()
for i in range(temp.shape[1]) :
    if(temp.iloc[:,i].dtypes == 'object') :
        temp.iloc[:,i] = pd.Categorical(temp.iloc[:,i])
        temp.iloc[:,i] = temp.iloc[:,i].cat.codes
SampleData = temp

#####################################
#  2.1.4 outlier analysis          #
#####################################
#####################################
# user defined function that will plot boxplot and histogram for four columns -> see output
def hist_and_box_plot(col1, col2, col3, data, bin1=30, bin2=30, bin3=30, bin4 = 30, sup =""):
    fig, ax = plt.subplots(nrows = 2, ncols = 3, figsize= (14,6))
    super_title = fig.suptitle("Boxplot and Histogram: "+sup,fontsize='x-large')
    plt.tight_layout()
    sns.boxplot(y = col1, data = data, ax = ax[0][0])
    sns.boxplot(y = col2,data = data, ax = ax[0][1])
    sns.boxplot(y = col3, data = data, ax = ax[0][2])
    sns.distplot(data[col1], ax = ax[1][0], bins = bin1)
    sns.distplot(data[col2], ax = ax[1][1], bins = bin2)
    sns.distplot(data[col3], ax = ax[1][2], bins = bin3)
    fig.subplots_adjust(top = 0.90)
    plt.show()
# plotting boxplot and histogram for our numerical variables
hist_and_box_plot('CountA', 'CountB', 'CountC', bin1 = 10, data = SampleData)

#####################################
#  2.1.4 Feature Engineering       #
#####################################

# Creating hour data variable
from datetime import datetime
SampleData['hour'] = temp1['Date'].dt.hour

# user defined function to plot bar plot of a column for each y i.e. y1 and y2 wrt
# unique variables of each x i.e. x1 and x2
# X1 and X2 would be categorical variable, y1 and y2 would be continuous
# this funciton will plot barplot for y1 column for each unique values of x1 and
# will do barplot for y2 for each unique values of x2 and method could be mean,sum etc.
see output
def plot_bar(x1, y1,x2, y2, method = 'sum'):
    fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize= (12,4), squeeze=False)
    super_title = fig.suptitle("Bar Plot ",  fontsize='x-large')
    if(method == 'mean'):
        gp = SampleData.groupby(by = x1).mean()
        gp2 = SampleData.groupby(by = x2).mean()
    else:
        gp = SampleData.groupby(by = x1).sum()
        gp2 = SampleData.groupby(by = x2).sum()
    gp = gp.reset_index()
    gp2 = gp2.reset_index()
    sns.barplot(x= x1, y = y1, data = gp, ax=ax[0][0])
    sns.barplot(x= x2, y = y2, data = gp2, ax=ax[0][1])
    fig.subplots_adjust(top = 0.90)
    plt.show()


# plotting barplot for count i.e. cnt wrt to yr and month
```

```
plot_bar('FeedBack', 'CountA', 'hour', ‘CountA')

# Creating categorical variable time_of_day from hour data variable
for i in range(SampleData.shape[0]):
    if SampleData.ix[i,'hour'] < 12 :
        SampleData.ix[i,'time_of_day'] = 'Morning'
    elif SampleData.ix[i,'hour'] <= 17 :
        SampleData.ix[i,'time_of_day'] = 'Afternoon'
    else :
        SampleData.ix[i,'time_of_day'] = ‘Night'

plt.figure()
plt.hist(SampleData['time_of_day'], alpha = 1, edgecolor = 'black')
plt.ylabel('Count of time_of_day')
plt.xlabel(‘time_of_day')

SampleData.loc[:,'time_of_day'] = pd.Categorical(SampleData.loc[:,'time_of_day'])
SampleData.loc[:,'time_of_day'] = SampleData.loc[:,’time_of_day'].cat.codes

# Dummy Variable Creation
num_var = ['CountA','CountB', 'CountC','hour']
cat_var = ['Type'] # Dropped as per chi-sq test ['User','time_of_day']
df1 = SampleData.applymap(int)
for i in cat_var:
    df1[i].astype('str')
temp1 = pd.DataFrame(df1['FeedBack'])
temp1 = temp1.join(SampleData[num_var])
for i in cat_var:
    d = pd.get_dummies(df1[i],prefix = i)
    temp1 = temp1.join(d)
data_hotencod = temp1
data_hotencod.shape,data_hotencod.columns


# let us look at correlation plot for each numerical variables -> see output
num_var = ['CountA','CountB','CountC','hour']
sns.pairplot(data_hotencod[num_var])

#correlation plot-> see output
num_var = ['CountA','CountB','CountC','hour']
data_corr = SampleData.loc[:,num_var]
#f,ax = plt.subplots(figsize = (7,5))
corr = data_corr.corr()

colormap = plt.cm.RdBu
plt.figure(figsize=(15,15))
plt.title('Pearson Correlation of Features', y=1.0, size=10)
sns.heatmap(data_corr.corr(),linewidths=0.2,vmax=1.0, square=True, cmap=colormap,
linecolor='white', annot=True)

# checking dependency between FeedBack and independent variable (category)
see output
cat_var = ['Type','User','time_of_day']
from scipy.stats import chi2_contingency
print("Chi-square - test of independence")
print("===================================")
for i in cat_var:
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(SampleData['FeedBack'], SampleData[i]))
    print("p-value between FeedBack and {}".format(i))
    print(p)
    print('----------------------------')

# checking independency between independent variables-> see output
chi2, p, dof, ex = chi2_contingency(pd.crosstab(SampleData['Type'],
                                                SampleData['User']))
print("p-value between 'Type' and 'User'")
print(p)
print('----------------------------')

# checking independency between independent variables
chi2, p, dof, ex = chi2_contingency(pd.crosstab(SampleData['Type'],
                                               SampleData['time_of_day']))
print("p-value between 'Type'  and 'time_of_day'")
print(p)
print('----------------------------')

# checking independency between independent variables
chi2, p, dof, ex = chi2_contingency(pd.crosstab(SampleData['User'],
                                               SampleData['time_of_day']))
print("p-value between 'User'  and 'time_of_day'")
print(p)
print('----------------------------')

# checking importance of feature-> see output
drop_col = ['User','time_of_day','FeedBack']
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier(n_estimators=200)
X = SampleData.drop(columns= drop_col)
y = SampleData['FeedBack']
clf.fit(X, y)
imp_feat = pd.DataFrame({'Feature': SampleData.drop(columns=drop_col).columns,
                          'importance':clf.feature_importances_})
imp_feat.sort_values(by = 'importance', ascending=False).reset_index(drop = True)

# Checking VIF values of numeric columns-> see output
from statsmodels.stats.outliers_influence import variance_inflation_factor as vf
from statsmodels.tools.tools import add_constant
numeric_df = add_constant(SampleData[num_var])
vif = pd.Series([vf(numeric_df.values, i) for i in range(numeric_df.shape[1])],
                 index = numeric_df.columns)
vif.round(1)



# splitting in X and y for train and test
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_hotencod.iloc[:,1:8],data_hotencod.iloc[:,0],
test_size=0.2, random_state=42)
```

```
###########################################
#                                                         #
#                                                         #
#   2.2.2 Building Classification models   #
#                                                        #
#                                                        #
###########################################

# making general function to fit and predict result (Confusion Matrix)
# and performance (K-fold CV) and to not to repeat code everytime
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score
def fit_predict_show_performance(classifier, X_train, y_train):
    '''
    this function will fit on data passed in argument then it will predict on
    X_test datasetand then will calculate the 10 fold CV accuracy score and then will
    generate classification report and confusion matrix based on prediction and y_test
    it will only print result, to get all calculated result, uncomment last line and
    call it like below example:
    y_pred, cr, cm = fit_predict_show_performance(classifier, X_train, y_train)
    '''
    # fitting model
    classifier.fit(X_train, y_train)
    prediction = classifier.predict(X_test)
    # getting K-fold CV scores for K = 10
    ten_performances = cross_val_score(estimator=classifier,X=X_train,y=y_train,cv=10)
    k_fold_performance = ten_performances.mean()
    print("K-fold cross validation score of model for k = 10 is :")
    print(k_fold_performance)
    print("=====================================")
    print("====== Classification Report ======= ")
    cr = classification_report(y_test,prediction)
    print(cr)
    print("====== Confusion matrix ======= ")
    cm = confusion_matrix(y_test,prediction)
    print(cm)
    #return [prediction, cr, cm]

########################
#  Logistic Regression  #
########################-> see output
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
fit_predict_show_performance(classifier, X_train, y_train)

########################
#       KNN             #
########################-> see output
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski',p =2)
fit_predict_show_performance(classifier, X_train, y_train)
```

```
#########################
#     Naive Bayes       #
#########################-> see output
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
fit_predict_show_performance(classifier, X_train, y_train)

#########################
#    Decision Tree      #
#########################-> see output
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=1)
fit_predict_show_performance(classifier, X_train, y_train)


#########################
#    Random Forest      #
#########################-> see output
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',
                                    random_state=1)
fit_predict_show_performance(classifier, X_train, y_train)


############################################
#                                       #
#                                       #
#       Hyperparameter tuning              #
#                                       #
#                                       #
############################################
#########################################
#                                    #
# tuning decision tree for dataset   #
#                                    #
#########################################

# hyperparameter tuning for Decision tree classifier
from sklearn.model_selection import GridSearchCV
classifier = DecisionTreeClassifier(random_state=1)
params = [{'criterion':['entropy', 'gini'],
         'max_depth':[6,8,10,12,20],'class_weight':['balanced',{0:0.45, 1:0.55},
                    {0:0.55,1:0.45},{0:0.40,1:0.60}],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=classifier, param_grid=params,
                      scoring = 'f1', cv = 10, n_jobs=-1)

# tuning Decision Tree for dataset-> see output
grid_search = grid_search.fit(X_train, y_train)
grid_search.best_params_


#Decision tree classifier
#from sklearn.tree import DecisionTreeClassifier-> see output
```

```
classifier = DecisionTreeClassifier(criterion = 'entropy',
                                     class_weight='balanced',max_depth=6,
                                     random_state=1)
fit_predict_show_performance(classifier, X_train, y_train)

############ Hyperparameter tuning for Random Forest #############
# Grid search for finding best parameter for random_forest -> see output
classifier = RandomForestClassifier(random_state=1)
params=[{'criterion':['entropy', 'gini'],'n_estimators':[800,1000],
        'max_depth': [8, 10, 12], 'class_weight':['balanced', {0:0.45, 1:0.55},
                   {0:0.55, 1:0.45}],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=classifier, param_grid=params,
                      scoring = 'f1', cv = 10, n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
grid_search.best_params_

# tuned randomforest model -> see output
classifier = RandomForestClassifier(n_estimators = 800, criterion = 'entropy',
                                     class_weight='balanced',max_depth=8,
                                     random_state=1)
fit_predict_show_performance(classifier, X_train, y_train)


############################################
#                                          #
#    SMOTE + Tomek (Oversampling)          #
#         Balancing Target                 #
#                                          #
############################################

# resmapling data -> see output
from imblearn.combine import SMOTETomek
smt = SMOTETomek()
#X_resampled, y_resampled = smt.fit_sample((X_train), (y_train))
X_resampled, y_resampled = smt.fit_sample(data_hotencod.iloc[:,1:8], data_hotencod.iloc[:,0])

# checking shape of data after resampling
print(X_resampled.shape)
print(y_resampled.shape)
print("class proportion")
print(pd.Series(y_resampled).value_counts(normalize = True))

# tuning Random forest model for resampled data-> see output
classifier = RandomForestClassifier(random_state=1)
params = [{'criterion':['entropy','gini'],'n_estimators':[600, 800, 1000],
         'max_depth': [24, 26, 28], 'random_state' :[1],
         'class_weight':['balanced', {0:0.45, 1:0.55},{0:0.55, 1:0.45}]}]
grid_search = GridSearchCV(estimator=classifier, param_grid=params,
                      scoring = 'f1', cv = 10, n_jobs=-1)
grid_search = grid_search.fit(X_resampled, y_resampled)
grid_search.best_params_


# building Random Forest model on tuned parameter-> see output
classifier = RandomForestClassifier(n_estimators = 600, criterion = 'entropy',
                                     class_weight='balanced',
                                     max_depth=24,random_state=1)
fit_predict_show_performance(classifier, X_resampled, y_resampled)
```

References

https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification- problems/
http://contrib.scikit-learn.org/imbalanced- learn/stable/auto_examples/combine/plot_smote_tomek.html