

Manage People Supply Chain Capgemini Hackathon

Abhishek H
abhishekhupele@gmail.com
8989935603
13 Oct, 2018

Contents

Introduction	1
1 Problem Statement	1
1.1 Data	4
Methodology	5
2. Build a Predictive Demand Model	6
2.1 Data Preprocessing: (Exploratory Data Analysis)	6
2.1.1 Understanding the Data	6
2.1.2 Missing value analysis	10
2.1.3 Visualising Data	11
2.1.4 Outlier Analysis:	11
Feature selection:	13
2.1.8 Data After EDA	17
2.1.9 Feature Engineering:	18
2.2 Modeling	20
2.3 Model Performance Comparison	22
3. Plan the optimized supply needed per month for the next 12 months.	23
3.1 Objective & Constraints	23
3.2 Overall Approach	24
4. Simulation	26
Annexure 1 : Flowchart symbols	27
Conclusion	28
3.1 Create the summary table (DA1.png):	28
3.2 Inference reasoned out from the dataset	28
3.3 Final Model and Training Dataset	28
3.4 End Notes	29
Complete Python code	30
References	42

Chapter 1

Introduction

InterstellarX Inc (IXI) is a company engaged in providing Strategy, Technology, Data sciences and Creative Design services to large global clients.

The core business entails revenue generation by providing the **right** resources to clients, across the world, for specific projects, based on service requirements.

1 Problem Statement

One of the most critical business compulsions as well as challenges for IXI, is to manage their People Supply Chain – by mapping the Demand for resources (A resource is tagged by metadata such as primary skillset, location, qualification and experience level etc See attached Demand.xls) to the Net Supply of resources (from various Internal and External hiring channels, keeping attrition in view).

IXI maintains, and periodically replenishes a **Bench** (Inventory) of resources depending upon forecasted demand for the next few quarters. There is a steep cost constraint associated with maintaining this bench (On average \$685 per person per month) and hence the company needs to ensure demand forecast is as accurate as possible, so Supply can be planned accordingly. (Maintaining a bench implies absorbing the cost of a resource who is not billable, and hence is an unproductive cost to the company till he/she is allocated to a billable project)

You are the lead data scientist for a central analytics team that has been assigned the task of managing this Demand and Supply Equilibrium by forecasting Demand with a 90% accuracy rate, and then optimizing Supply to meet the forecasted demand.

Additionally, company leadership wants to build a machine learning based front-end application that allows top management to change variables and see how that affects demand-supply and consequently the Profits and losses for the company.

You need to:

1. Build a Predictive Demand Model – Can be trend/time series based **OR** causal. Provide a reason for choosing the one that you use.
2. Basis $n+2$ month forecasted demand above, plan the **optimized** supply needed per month for the next 12 months.
3. Based on the associated costs constraints given in Table 1 below, create a simple simulation that showcases the Net loss of business and/or Net additional cost if variable factors are changed and demand and supply changes as a consequence.

Table 1: Points/constraints to consider, when planning the Optimized Supply for a particular month:

Supply

Supply needs to be planned 2 months in advance, hence month N supply needs to be mapped to month N+2 demand forecast i.e Supply (N) = Demand (N+2)

In any month **if Demand exceeds Supply**, the estimated per resource, per month loss of business revenue to the company is \$900. (This is not a cash loss but an Opportunity Cost. Accordingly assume \$900 per month is also the average revenue for a billable resource)

Total budget for maintaining the Bench for the current year is \$5.76 Mn.

Average cost per resource per month is \$685.

Current Bench strength is 400, implying that annualized Bench budget consumption is at \$3.288 Mn on day 1 of the year.

End of year average Bench cost cannot exceed total budget of \$5,76 Mn.

Assume Average annual attrition of 20% of total headcount

Total headcount at the beginning of the year is 10000 and cannot exceed 12000 at the end of the year. Assume that everyone who is not on Bench is a billable/billed resource.

Assume you are starting this project at the beginning of the year.

Total Headcount at a point of time = Bench + New External Hires – Attrition

Assume there are only 2 supply channels:

- 1) Internal Bench available immediately (Preferable if a match is found)
- 2) External Hire available after a 2 month lag

Assume that once billed a resource stays with the same account forever and does not come back into the bench or move to any other project

While having a strong bench ensure the least loss of business, these losses are notional and not real losses. In contrast, if demand falls short the company has to incur the additional resource costs till such time that excess bench can be placed in a billing role. Hence there is a preference for keeping the bench minimal and optimized to meet the certain demand.

Assume there are no other associated costs

Notes:

- 1) For all models ensure training and testing sets are divided in an 80:20 split
- 2) Clearly state all assumptions and the rationale for selecting the algorithm you have used to build a model.
- 3) Explain your results, with a view to helping a non-technical person understand all insights.
- 4) Demonstrate accuracy of both training set as well as Out-of-Sample tests.
- 5) **Development environment on R/Python is acceptable.**

Data set

[Download the sheet here](#) containing Headcount and Demand Trend Last of last year.

Judging Criteria

You will be judged on your proficiency in Machine Learning, Analytics, Strategic Approach to the problem and finally presentation of your output.

Deliverables

- Source code of all three tasks along with training data and verification instructions
- Presentation of strategic approach and your output

Expected Deliverables**Upload Files**

Source code of all three tasks along with training data and verification instructions.

Drag and drop or upload your submission file below

Upload File

Presentation of strategic approach and your output. *

Drag and drop or upload your submission file below

Upload File

1.1 Data

Data is an excel file with two sheet :

(i) Headcount

Headcount								
Region	Employee Code	Last Name	Local Date of Joining	Designation	Status	Market Unit	SkillList	Location
US	45149	AALLURI	10/31/16	CONSULTANT	Billable	BANK	Salesforce (Func	Bangalore
IN	47875	AARATTUK	11/20/14	SENIOR SOFT	Billable	BANK	Amazon Web ser	Bhubaneswar
IN	90386	AASHIK	12/15/14	ASSOCIATE CO	Billable	BANK	Amazon Web ser	Pune
IN	41161	Aastha	2/26/15	ASSOCIATE CO	Billable	BANK	Test Automation	Bhubaneswar
IN	92293	ABBAN	12/2/13	CONSULTANT	Billable	BANK	PMO (Account)	Noida
IN	79912	ABBANOLL	1/19/12	ASSOCIATE CO	Billable	BANK	Salesforce (Func	Gurgaon

'Region' - Region,
 'Employee Code' - Unique Employee Id
 'Last Name' - Employee Last Name
 'Local Date of Joining' - Date of Joining Firm
 'Designation' - Designation of Employee
 'Status' - Billable or Bench Status
 'Market Unit' - Industry Sector
 'SkillList' - Technical Skill Set
 'Location' - City of Employee Residence

(ii) Demand Trend Last Year

Demand Trend of Last Year							
Month DD Raised	No. of FTE Request Raised	SkillList	Location	Experience Grade	Practice	Skill Group	Demand Source
May	4	Salesforce (Functional)	Bangalore	A4	BANK	(F) IT Operat	Account
October	2	Amazon Web services - Bu	Bhubanesw	B2	M&FT	(F) IT Operat	Account
March	2	Amazon Web services - Bu	Pune	A3	TEST		Support
October	1	Test Automation	Bhubanesw	B2	M&FT	(F) IT Operat	Account
June	2	PMO (Account)	Noida	A5	I&D	(D) Banking S	Proactive

'Month DD Raised' - Month of Demand/Requirement request raised
 'No. Of FTE Request Raised' - Number of Employees Required
 'SkillList' - Technical Skill Set
 'Location' - Location of Client
 'Experience Grade' - Years of Experience(Categorical)
 'Practice' - Billable or Bench Status
 'Market Unit' - Industry Sector
 'Skill Group' - Technical Skill Category
 'Demand Source' - Type of Demand/Department

Chapter 2

Methodology

We need to:

- 1. Build a Predictive Demand Model – with 90% accuracy.** Based on Demand Trend Last Year
- 2. Basis n+2 month forecasted demand above, plan the optimized supply needed per month for the next 12 months.**

$$\text{Supply}(N) = \text{Demand}(N+2)$$

$$\text{Total Head Count} < 12,000$$

$$\text{Total Bench Budget} < \$ 5.76 \text{ Million}$$

$$\text{Avg. Annual Attrition} = 20\% \text{ of Total Head Count}$$

$$\text{Bench Cost/Month} = \$685$$

$$\text{Opportunity Cost (Unmet Demand/Consultant/Month)} = \$900$$

$$\text{Available Resource Headcount at a point of time} = \text{Bench} + \text{New External Hires} - \text{Attrition}$$

Once billed a resource stays with the same account forever and does not come back into the bench or move to any other project.

$$\text{Other associated cost(or expenses)} = 0$$

- 3. Based on the associated costs constraints given in Problem Statement (Table 1) create a simple simulation that showcases the Net loss of business and/or Net additional cost if variable factors are changed and demand and supply changes as a consequence.**

2. Build a Predictive Demand Model

2.1 Data Preprocessing: (Exploratory Data Analysis)

- Demand Trend Last Year

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 visualisation.

2.1.1 Understanding the Data

First, we will look at our ‘Demand Trend for Last Year’ data and datatype:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1023 entries, 0 to 1022
Data columns (total 8 columns):
Month DD Raised      1023 non-null object
No. of FTE Request Raised  1023 non-null int64
SkillList            1023 non-null object
Location             1023 non-null object
Experience Grade      1023 non-null object
Practice             1023 non-null object
Skill Group          981 non-null object
Demand Source        1023 non-null object
dtypes: int64(1), object(7)
memory usage: 64.0+ KB
```

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

Categorical Variable:

'Month DD Raised', 'SkillList', 'Location', 'Experience Grade', 'Practice', 'Skill Group',
'Demand Source'

Continuous Variable / Target variable :

'No. of FTE Request Raised'

Let us now analyse our numerical variable :

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

	No. of FTE Request Raised
count	1023.000000
mean	2.482893
std	1.118012
min	1.000000
25%	2.000000
50%	2.000000
75%	4.000000
max	4.000000

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

Analysis:

We target variable as continuous data.

As we can observe from above table, all dependent variable are between 1 and 4.

Checking categorical data Now:

Counting of each unique value in each categorical variable.

Value counts of Categorical Columns

Month DD Raised ## :-

March 196
October 188
August 162
June 150
January 90
December 87

February 86
May 64
Name: Month DD Raised, dtype: int64

SkillList ## :-

Java
71
Ab Initio
50
Business Analytics,Cards Issuing
28
MS/Windows Dev OPS
27
Angular.js
26
.....

Location ## :-

Pune 196
Bhubaneswar 188
Gurgaon 162
Noida 150
Chennai 90
Hartford 87
Hyderabad 86
Bangalore 64
Name: Location, dtype: int64

Experience Grade ## :-

A3 196
B2 188
B1 162
A5 150
A1 90
C1 87
A2 86
A4 64
Name: Experience Grade, dtype: int64

Practice ## :-

TEST 196
M&FT 107
INS 96
DCX 90
ERP 87
CFS 86

```

ADM      83
Cards    81
I&D      67
GP       66
BANK     64
Name: Practice, dtype: int64

```

Skill Group ## :-

```

(T) Enterprise Content Management (ECM)          154
(D) Banking Sector                               131
(T) Mainframe Development Technologies           103
(T) Java Development Technologies                71
(T) Card packages                               62
(F) IT Operations                               55
(D) Cards                                         38
(T) Testing Tools / Testing Automation / Performance Testing Tools  37
(T) Project Tools, Methodologies and Frameworks  31
(T) Microsoft Development Technologies           30
Name: Skill Group, dtype: int64

```

Demand Source ## :-

```

Proactive  610
Account    310
Support    103
Name: Demand Source, dtype: int64

```

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

Analysis:

‘Month DD Raised ‘ - Demand Values are missing for months April, July, Sep, Nov .It should be checked to understand the reasons or gap in data collection.

‘Location’ - 1 is out of India, rest 7 are within India

Assumption : Location is not a constraint , Employees can be relocated to match demand with nil expenses (Other associated cost(or expenses) = 0). Though first priority will be given to same location available employee.

Experience Grade : Need definition for levels

Assumption : Experience is ordinal variable, with ageing in organisation as below

```

A5. Oldest
A4 Older
A3 Old
A2 Younge
A1 Younger
B2 Youngest
B1 Teenage

```

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:

Attributes	Cnt of Miss	% Miss
Skill Group	42	4.105572
Month DD Raised	0	0
No. of FTE Request Raised	0	0
SkillList	0	0
Location	0	0
Experience Grade	0	0
Practice	0	0
Demand Source	0	0

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

Analysis:

On Analysis of Missing Values of 'Skill Group' attribute, we see that there is some indication from 'Skill List' and 'Practice' but it is not definite pattern for imputation manually. We have not considered 'Demand Source' as it seems indicating Application Status/hiring status and is not related to skillset required.

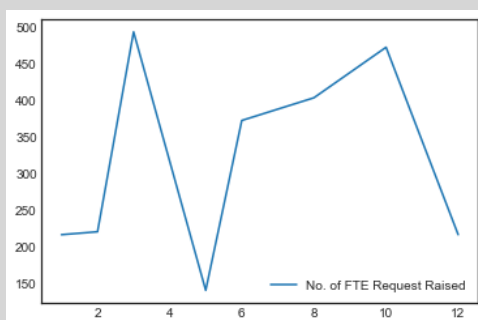
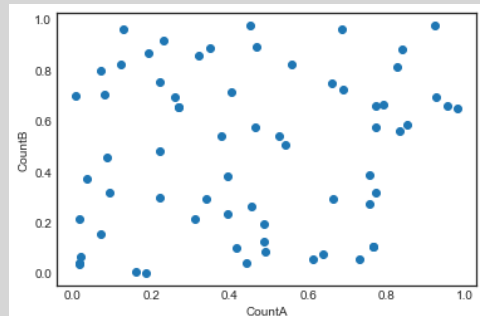
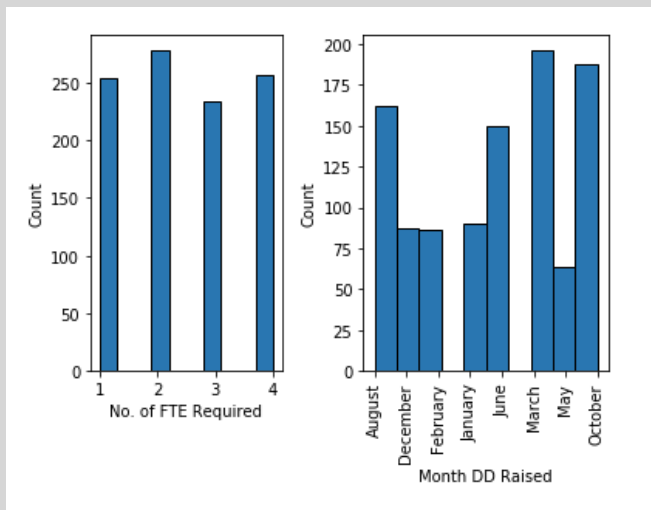
We will use 'KNN' method to impute Missing Values. As 'Mode' method gives NaN as mode value.

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

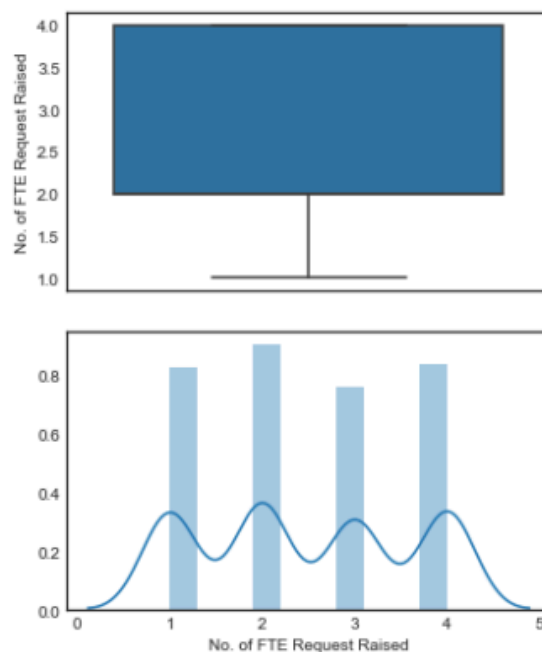
Q3 > 75% of data are less than or equal to this value

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 .

Feature selection:

For feature selection we will see if there is dependency between target and independent variable by ANOVA analysis, further we will check for multicollinearity between our categorical independent variable.

Anova test:

An ANOVA test is a way to find out if survey or experiment results are significant. In other words, they help you to figure out if you need to reject the null hypothesis or accept the alternate hypothesis. Basically, you're testing groups to see if there's a difference between them. Examples of when you might want to test different groups

When there are two or more categorical factors in our model, we again may want to test various single degrees-of-freedom hypotheses that compare various levels of the two or more factors in the model. As with the one-way ANOVA model, how you parameterize your two-way or higher model affects how you go about performing individual tests. ANOVA assume continuous values in the dependent variable, but categorical variables as the independent variables

P-value

To determine whether any of the differences between the means are statistically significant, compare the **p-value** to your significance level to assess the null hypothesis.

The null hypothesis states that the population means are all equal.

Usually, a significance level of 0.05 works well.

If the p-value is greater than the significance level, you do not have enough evidence to reject the null hypothesis that the population means are all equal. Verify that your test has enough power to detect a difference that is practically significant.

```
#### Anova Test Month DD Raised ####
Null Hypothesis Accepted : means are all equal
              df      sum_sq  mean_sq      F
PR(>F)
C(Month)      7.0      6.636194  0.948028  0.75719
0.623459
Residual 1015.0 1270.814441  1.252034      NaN
NaN
#### Anova Test Skill List ####
Null Hypothesis Accepted : means are all equal
              df      sum_sq  mean_sq      F
PR(>F)
C(SkillList) 221.0 295.014063  1.334905  1.088375
0.207753
Residual    801.0 982.436572  1.226513      NaN
NaN
#### Anova Test Location ####
```

```

Null Hypothesis Accepted : means are all equal
              df          sum_sq    mean_sq          F
PR(>F)
C(Location)      7.0          6.636194    0.948028    0.75719
0.623459
Residual      1015.0    1270.814441    1.252034          NaN
NaN
#### Anova Test Experience ####
Null Hypothesis Accepted : means are all equal
              df          sum_sq    mean_sq          F
PR(>F)
C(Experience)      7.0          6.636194    0.948028    0.75719
0.623459
Residual      1015.0    1270.814441    1.252034          NaN
NaN
#### Anova Test Practice ####
Null Hypothesis Accepted : means are all equal
              df          sum_sq    mean_sq          F
PR(>F)
C(Practice)     10.0          8.037007    0.803701    0.640725
0.77952
Residual     1012.0    1269.413628    1.254361          NaN
NaN
#### Anova Test Skill Group ####
Null Hypothesis Accepted : means are all equal
              df          sum_sq    mean_sq          F
PR(>F)
C(SkillGroup)   51.0          73.133862    1.433997    1.156184
0.21472
Residual      971.0    1204.316774    1.240285          NaN
NaN
#### Anova Test DemandSource ####S
Null Hypothesis Accepted : means are all equal
              df          sum_sq    mean_sq          F
PR(>F)
C(DemandSource)    2.0          5.338025    2.669012    2.140056
0.118176
Residual      1020.0    1272.112611    1.247169          NaN
NaN

```

Let us analyze for categorical variables also:

We have 7 categorical variable. For getting dependency between categorical variables we would use chi-sq test of independence test.

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.

Null Hypothesis for Chi-square test of Independence:

Two variables are independent.

Alternate Hypothesis:

Two variables are not independent

	Month DD Raised	SkillList	Location	Experience Grade	Practice	\
Month DD Raised	-	0.51	0.0	0.0	0.0	
SkillList	0.51	-	0.51	0.51	0.18	
Location	0.0	0.51	-	0.0	0.0	
Experience Grade	0.0	0.51	0.0	-	0.0	
Practice	0.0	0.18	0.0	0.0	-	
Skill Group	0.547	1.0	0.547	0.547	0.331	
Demand Source	0.395	0.001	0.395	0.395	0.64	

	Skill Group	Demand Source
Month DD Raised	0.547	0.395
SkillList	1.0	0.001
Location	0.547	0.395
Experience Grade	0.547	0.395
Practice	0.331	0.64
Skill Group	-	0.522
Demand Source	0.522	-

[** 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.

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

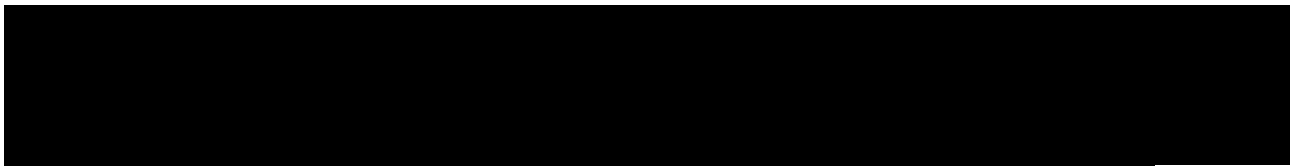
amount. And if we drop holiday, that information is also included in working day.

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 data-points would not be showing collinearity and at that datapoint there could be abrupt difference in 'No. of FTE Requirement' count column which is very important information for our model.

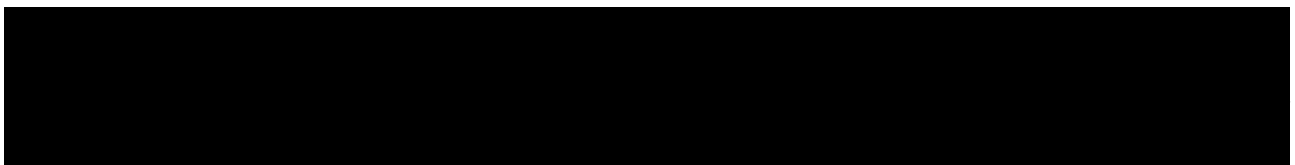
We will drop only holiday column as we have working day column which has information of holiday also, that is why on dropping holiday we are not losing our accuracy. Also in while getting important features we are getting last ranking for holiday column with very less importance.

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

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:



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.



Let us check now important feature of our dataset:

Feature	importance
SkillList	0.42514
Skill Group	0.334707
Practice	0.067072
Demand Source	0.049048
Experience Grade	0.047142
Location	0.041638
Month DD Raised	0.035252

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

Feature Scaling: Not Required

2.1.8 Data After EDA

```
In [33]: 1 Demand_LY.head()
```

```
Out[33]:
```

	Month DD Raised	No. of FTE Request Raised	SkillList	Location	Experience Grade	Practice	Skill Group	Demand Source
0	6	4	175	0	3	1	16.0	0
1	7	2	17	1	6	9	16.0	0
2	5	2	17	7	2	10	30.0	2
3	7	1	195	1	6	9	16.0	0
4	4	2	141	6	4	7	0.0	1

We will make dummy variables for categorical variables.

One hot encoding of factor variables

One hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1

I have done one hot encoding of categorical data

So the number of binary columns depend on number of factor levels in the raw column

Now all the data is converted to numeric features

By one hot encoding we will convert the factor to binary columns which is numeric in nature

For example

Color		Red	Yellow	Green
Red		1	0	0
Red		1	0	0
Yellow		0	1	0
Green		0	0	1
Yellow		0	0	1

```
1 data_hotencod.head()
```

	No. of FTE Request Raised	Month DD Raised_0	Month DD Raised_1	Month DD Raised_2	Month DD Raised_3	Month DD Raised_4	Month DD Raised_5	Month DD Raised_6	Month DD Raised_7	SkillList_0	...	Skill Group_45.0	Skill Group_46.0	Skill Group_47.0	Skill Group_48.0
0	4	0	0	0	0	0	0	1	0	0	...	0	0	0	0
1	2	0	0	0	0	0	0	0	1	0	...	0	0	0	0
2	2	0	0	0	0	0	1	0	0	0	...	0	0	0	0
3	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0
4	2	0	0	0	0	1	0	0	0	0	...	0	0	0	0

5 rows x 313 columns

2.1.9 Feature Engineering:

Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature Engineering is an art.

Steps which are involved while solving any problem in machine learning are as follows:

- Gathering data.
- Cleaning data.
- **Feature engineering.**
- Defining model.
- Training, testing model and predicting the output.

I have used selectKbest in python

SelectKBest selects the top k features that have maximum relevance with the target variable. It takes two parameters as input arguments, "k" (obviously) and the score function to rate the relevance of every feature with the target variable. For example, for a regression problem, you can supply

Select features according to the k highest scores. The score function must return an array of scores, one for each feature (additionally, it can also return p-values, but these are neither needed nor required). SelectKBest then simply retains the first k features with the highest scores.

Principal component analysis:

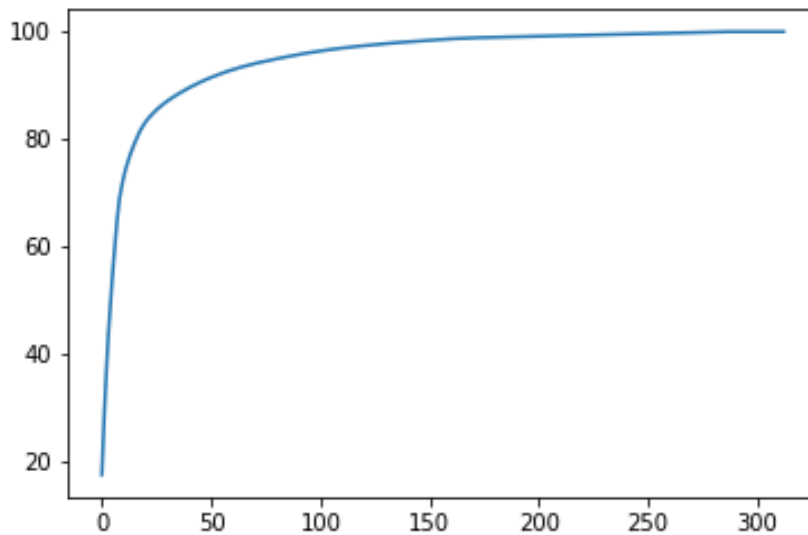
As we have more features in the data(new columns obtained from one hot encoding) the dimension will be large.

So i have used Principal component analysis to reduce the dimension of the data keeping the variance unchanged

Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

Principal component analysis is a technique for feature extraction — so it combines our input variables in a specific way, then we can drop the least important variables while still retaining the most valuable parts of all of the variables! As an added benefit, each of the

new variables after PCA are all independent of one another. This is a benefit because the assumptions of a linear model require our independent variables to be independent of one another. If we decide to fit a linear regression model with these new variables (see principal component regression below), this assumption will necessarily be satisfied.



From the above plot ~50 variables clearly explains the almost 90% of variance in the data

So PCA has reduced 300 variables to 50 without compromising the variance in the data

Sampling methods

K-fold repeated CV

- k-fold cross-validation randomly divides the data into k blocks of roughly equal size. Each of the blocks is left out in turn and the other k-1 blocks are used to train the model. The held out block is predicted and these predictions are summarized into some type of performance measure (e.g. accuracy, root mean squared error (RMSE), etc.). The k estimates of performance are averaged to get the overall resampled estimate. k is 10 or sometimes 5.
- Repeated k-fold CV does the same as above but more than once. For example, five repeats of 10-fold CV would give 50 total resamples that are averaged. Note

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 ('No. of FTE Request Raised') of data_hotencod
- X_test : containing independent variable (part of Demand_LY), used for testing model
- y_test: containing target variable(part of Demand_LY), , used for testing model
- fit_predict_show_performance: user-defined function, which will fit our model on training set, and will calculate and print k-fold (10-fold) cross validation score for explained_variance , and then will make prediction on training and test dataset and will print explained_variance for both training and test dataset.

Models and performance of models :

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

Linear Regression::

RMSE : 1.190850642312722

Ridge Regression :

RMSE : 1.1200419575291747

Model Score : -0.1105984787855201

Lasso Regression:

MSE : 1.0728421535294617

Model Score : -0.018967069423394545

Lars Regression :

RMSE : 4.499804332735489e+30

Model Score : -1.7925684545140844e+61

Decision Tree :

RMSE : 1.428797891457762

Model Score : -0.8072993469559722

Random Forest :

RMSE : 1.1232464262649058

Model Score : -0.11696247095559253

PCA results:

I have applied PCA to my data and it has given me very good results

Linear Regression PCA:

RMSE : 2.654005923569303

Ridge Regression PCA :

RMSE : 0.001977326264924357

Model Score : 0.9999968836977942

Lasso Regression PCA:

RMSE : 0.9021389687059362

Model Score : 0.35132108333650447

Lars Regression PCA:

RMSE : 1.6572307352378184

Model Score : 0.9999994549627828

Random Forest PCA:

RMSE : 1.6572089377669414

Model Score : 1.0

Decision Tree PCA:

RMSE : 0.0

2.3 Model Performance Comparison

Python results(without PCA)

Model	RMSE	R-Sqaue
Linear R	1.19	
Ridge R	1.12	-0.11
Lasso R	1	-0.018
Lars R	4.49	-1.79
Random Forest	1.12	-0.11
Decision Tree	1.42	-0.8

PCA results in python

Model	RMSE	R-Sqaue
Linear R	2.65	-
Ridge R	1.65	0.99
Lasso R	0.90	0.35
Lars R	4.49	-1.79
Random Forest	1.65	1
Decision Tree	0	-

So over all PCA has produced the best results for the above problem

Selecting the best fitted model:

Ridge and RSandom Forest using PCA is giving the best results

3. Plan the optimized supply needed per month for the next 12 months.

3.1 Objective & Constraints

Objective : $\text{Supply}(N) = \text{Demand}(N+2)$

Constraints :

Total Head Count < 12,000

Total Bench Budget < \$ 5.76 Million

Avg. Annual Attrition = 20% of Total Head Count

Bench Cost/Month = \$685

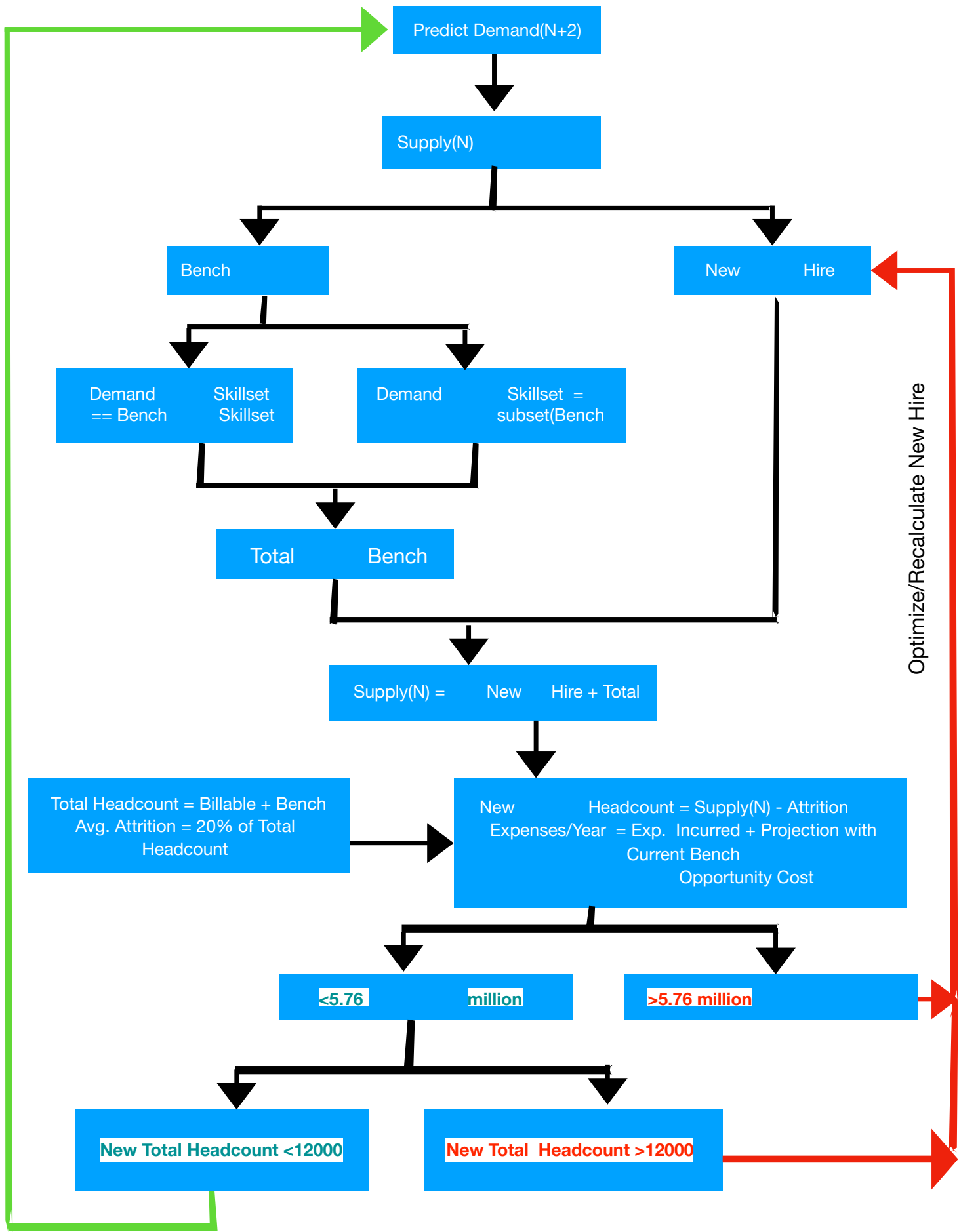
Opportunity Cost (Unmet Demand/Consultant/Month) = \$900

Available Resource Headcount at a point of time = Bench + New External Hires – Attrition

Once billed a resource stays with the same account forever and does not come back into the bench or move to any other project.

Other associated cost(or expenses) = 0

3.2 Overall Approach





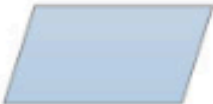


4. Simulation

= Bench * \$685 +incurred expenses
Opportunity Cost

Annexure 1 : Flowchart symbols

Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them. These are known as flowchart symbols.

The type of diagram dictates the flowchart symbols that are used. For example, a data flow diagram may contain an Input/Output Symbol (also known as an I/O Symbol), but you wouldn't expect to see it in most process flow diagrams.

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

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 function 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 dataset and 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                  #
#                                     #
```

```
#####
```

```
# resampling 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