HIRE HEROES USA CLIENT SERVICES: TERADATA 2019 DATA CHALLENGE

Team Members

Yash Tiwari A20110527

Subhash Daggubati A20090700

Anup Kumar Chittimalla A20171762

TABLE OF CONTENTS

Executive Summary

We are working on the Teradata 2019 Data Challenge where we are provided with multiple data sets and business questions from Hire Heroes USA: one of the non-profit partners of Teradata. Hire Heroes USA provides personalized job-search assistance to U.S military, veteran and military spouse community. In particular, its clients are aided in resume review, workshops, webinars, networking, mock- interviews, job-search etc.

For this data challenge, we have analyzed the data related to the 'Hire Heroes USA: Clients Services program' to provide valuable insights into how the clients demographic profile influences when a particular client registers for services and the likelihood that, a client completes a survey. These results help Hire Heroes USA to understand the patterns and behavior of its clients. Additionally, the results will help Hire Heroes USA to operate effectively and increases the usability of their system and help in better transition of its clients.

Statement of Scope

The dataset contains data of 1,32,430 records comprised of US military members, veterans and military spouses. It contains the clients data such as branch, pay grade, service time, spouse status etc. The following business questions will help in deriving insights about how to improve the enrollment process of Hire Heroes. The analysis can potentially save clients time and can significantly smoothen the process of hiring. This analysis also helps veteran service

organizations and federal agencies in better understanding of its clients (military, veteran and military spouse community).

- The relationship between a client's demographic profile (rank, branch, time in service etc.) and when that client registers for services?
- How are factors like rank, branch, spouse status etc. are affecting the client's likelihood to complete a survey?

Project Objectives

- By getting insights about the likelihood to complete a survey; different methods can be established that can help gathering information by Hire Heroes.
- The process of hiring can be improved to a great extent when hidden information about the behavior of the members would be unveiled. Changes can be implemented accordingly by knowing which ranks, branches registers for the services earlier or later that helps in preparing in advance and help queuing tasks.

Business question 1: The relationship between a client's demographic profile (rank, branch, time in service etc.) and when that client registers for services?
Target Variables:

- Joining_Period : which is Difference between military separation date and created date in Hire heroes USA

Predictor Variables

- Col : which is Administrative classification of ranks by pay grade and branch of military service
- Service_Branch__C : which is the Branch of military service (Army, Navy, etc.)

- Service_Period : which is the Time in military service in months (difference between separation date and entry date in military)
- Military_Spouse_Caregiver__C : which is True / False (if true, indicates job seeking client is a spouse of veteran / servicemember and does not have military service)

Business question 2: How are factors like rank, branch, spouse status etc. are affecting the client's likelihood to complete a survey?

Target Variables

- Alumni_Survey_Completed__c: True / False (if true, client responded to / completed alumni program survey)

Predictor Variables: (same as for business question 1)

- Col : which is Administrative classification of ranks by pay grade and branch of military service
- Service_Branch__C : which is the Branch of military service (Army, Navy, etc.)
- Service_Period : which is the Time in military service in months (difference between separation date and entry date in military)
- Military_Spouse_Caregiver__C : which is True / False (if true, indicates job seeking client is a spouse of veteran / servicemember and does not have military service)

Project Schedule

The team anticipates to complete the project in 89 days. The project meet is scheduled two times in a week. We generally discuss what each member worked on the week before and what

we should work on the current week. Individually, we need 4 to 5 hours to complete the weekly work. We don't work on weekends. If there is a holiday the day we were supposed to meet, we generally meet the day later and never delay on our weekly assignment.

TASKS	W1	February W2	February W3	W4	W1	March W2	March W3	March W4	W5	W1	April W2	April W3	W4	May W1
Dataset confirmation														
Project Proposal														
Scope Statement														
Data Access														
Data Consolidation														
Data Cleaning														
Data Transformation														
Data Reduction														
Data Statistics														
Data Dictionary														
Deliverable 1 - Submission														
Modeling Technique														
Build Model														
Assess Model														
Combine Deliverable 1 & Deliverable 2														
Final Report														
Final Presentation														
Presentation Record														
Final Deliverable														

Data Preparation

Data Access

The datasets were obtained through Teradata 2019 Data Challenge. The files are of type .csv

There are 13 datasets given as a part of the Challenge which sum to 764 MB. We are using only 1 dataset of 209 MB which contains all the information required for Client Services. We also required the grouping category name i.e military rank (Warrant Officer, Enlisted Members, Commissioned Officers) for different Branch (Army , Navy, Marines etc.) based on their pay-scale. The data for this was not available in the teradata data sets. The required military rank was available at https://militarybenefits.info/armed-forces-comparative-pay-grades-and-ranks/

So, we imported the data available at the above link into a csv from https://app.import.io/

website. The import process was an easy one. We started with what specific columns and rows we required and clicked on import and then downloaded.This was a 5 KB file. We later consolidated the columns in a new file along with the required columns for our analysis.

Data Consolidation

We used SalesForce_Contact.csv and MilitaryRank.csv files to consolidate the data. SalesForce_Contact.csv file contains the data of all the people who are associated with HireHerosUSA (Ex: Clients/Jobseekers, Volunteers, Donors, Transition Specialists). As our analysis is completely based on Clients/Jobseekers this data has been filtered based on Client__c=1. So this brings down the data to 105744 rows.

```
In [303]: clients = contact[contact["Client__c"]==1]


Out[304]: (105744, 391)
```

All the relevant columns that will help us define rank, branch, time in service, spouse status, when client registered and likelihood to complete Alumni Survey are selected from this data set.

```
In [306]: clients_req_col = clients.loc[:,["Id",
     ...:                                  "Client__c",
     ...:                                  "Date_of_Service_EntryNew__c",
     ...:                                  "Date_of_SeparationNew__c",
     ...:                                  "Service_Branch__c",
     ...:                                  "Service_Rank__c",
     ...:                                  "Military_Spouse_Caregiver__c",
     ...:                                  "CreatedDate",
     ...:                                  "Gender__c",
     ...:                                  "Race__c",
     ...:                                  "Alumni_Survey_Completed__c",
     ...:                                  "Active_Color__c"
     ...:                                  ]
     ...:                                ]
     ...:

In [307]:
```

Service_Rank column has the codes based on the levels of latest pay grade of the clients when they register at HireHerosUSA. To make it more readable by the business we have used external data source MilitaryRank.csv. Both the data sets have been merged based on the Ranks.

```
In [307]: Clients_Military=pd.merge(clients_req_col,military_data_req,
on='Service_Rank__c')
```

After this point we need two different data sets to answer these questions. Because the first question talks about when the client joins HireHerosUSA, in this case we can consider all the clients who are part of the system. But, for the second question we are talking about Alumni Survey which will be applicable to the clients who are already hired. i.e with Color_Code = Blue.

```
#Filter the clients to whom the survey has been sent.
clients_req_col_Alumni=clients_req_col_Alumni[clients_req_col_Alumni["Active_Color__c"]=='Blue']
```

```
In [310]: clients_req_col_Alumni.shape
Out[310]: (27375, 10)

In [311]:
```

Data Cleaning

Converting Dates

We have three Date Columns in our dataset(Service_Start_Date, Service_End_Date and Created Date). All these variables are in the object form. along with TimeStamp. But while calculating Service_Period (Time in service) and Joining_Period (When client registers with HireHerosUSA) we dont require the time stamp. So the time stamp portion of the Date string has been trimmed with the help of Pandas String functions (replace and split)

```
#Removing time stamp and converting datetime format
clients_req_col["Date_of_Service_EntryNew__c"] = clients_req_col["Date_of_Service_EntryNew__c"].str.replace(r' 0:00$', '')
clients_req_col["Date_of_Service_EntryNew__c"]
clients_req_col["EntryDate"]=pd.to_datetime(clients_req_col["Date_of_Service_EntryNew__c"], format = "%m/%d/%Y" ,infer_datetime_format=True,errors = 'coerce')
clients_req_col["EntryDate"].max()

clients_req_col["Date_of_SeparationNew__c"] = clients_req_col["Date_of_SeparationNew__c"].str.replace(r' 0:00$', '')
clients_req_col["Date_of_SeparationNew__c"]
clients_req_col["LastDate"]=pd.to_datetime(clients_req_col["Date_of_SeparationNew__c"], format = "%m/%d/%Y" ,infer_datetime_format=True,errors = 'coerce')
clients_req_col["LastDate"].max()
```

```
#Converting Created Date into Date format.
clients_req_col["CreatedDate"] = clients_req_col["CreatedDate"].str.split(" ",expand=True)[0]
#clients_req_col["CreatedDate"] = clients_req_col["CreatedDate"].str.replace(r' 0:00$', '')
clients_req_col["CreatedDate"]=pd.to_datetime(clients_req_col["CreatedDate"], format = "%m/%d/%Y" ,infer_datetime_format=True,errors = 'coerce')
clients_req_col["CreatedDate"].isnull().sum()
```

Deleting Erroneous Data:

When we are calculating the variable Service_Period (Time in Service) we saw some negative values. Which shows that the data has some incorrect entries where Service_Start_Date is later to the Service_End_Date. So this data has been eliminated deleted from the data set by checking for the rows with Start date later than End Date while doing this we need to make sure that we don't lose the spouse data as the date column may have null values in these columns.

```
#cleaning Erroneous Dates:
clients_req_col = clients_req_col[((clients_req_col["EntryDate"]<clients_req_col["LastDate"])
                  & (clients_req_col["EntryDate"].notnull()) & (clients_req_col["LastDate"].notnull()))
                  | clients_req_col["Military_Spouse_Caregiver__c"]==1]
clients_req_col.shape
```

Deleting the NaT values of Dates:

While transforming the Date Columns to DateTime the function pandas.to_datetime will return NaT values when the Date is greater than 2270-12-31. This would create Na values all the there Date columns after the transformation. These rows should be filtered out.

```
#cleaning Erroneous Dates:
clients_req_col = clients_req_col[((clients_req_col["EntryDate"]<clients_req_col["LastDate"])
                  & (clients_req_col["EntryDate"].notnull()) & (clients_req_col["LastDate"].notnull()))
                  | clients_req_col["Military_Spouse_Caregiver__c"]==1]
clients_req_col.shape
```

Checking Na values of other columns.

```
In [313]: clients_req_col.isnull().sum()/len(clients_req_col)*100
Out[313]:
Id                              0.000000
Client__c                       0.000000
Date_of_Service_EntryNew__c     0.000000
Date_of_SeparationNew__c        0.000000
Service_Branch__c               0.106598
Service_Rank__c                 0.476808
Military_Spouse_Caregiver__c    0.000000
CreatedDate                     0.000000
Gender__c                      22.881014
Race__c                        75.208874
Alumni_Survey_Completed__c      2.983290
Active_Color__c                 0.001441
EntryDate                       0.000000
LastDate                        0.000000
Service_Period                  0.000000
Joining_Period                  0.000000
dtype: float64
```

From the above table we can see that Gender and Race columns have more than 20% of the data so we can completely delete these columns from the Data.

```
In [316]: clients_req_col=clients_req_col.drop(["Gender__c","Race__c"],axis=1)
     ...: clients_req_col.isnull().sum()/len(clients_req_col)*100
Out[316]:
Id                              0.000000
Client__c                       0.000000
Date_of_Service_EntryNew__c     0.000000
Date_of_SeparationNew__c        0.000000
Service_Branch__c               0.106598
Service_Rank__c                 0.476808
Military_Spouse_Caregiver__c    0.000000
CreatedDate                     0.000000
Alumni_Survey_Completed__c      2.983290
Active_Color__c                 0.001441
EntryDate                       0.000000
LastDate                        0.000000
Service_Period                  0.000000
Joining_Period                  0.000000
dtype: float64
```

The rest of the variables have missing values of less than 3% so, we can delete the rows corresponding to these missing values.

```
In [317]:
clients_req_col=clients_req_col[clients_req_col["Service_Branch__c"].notnull()
     ...: & clients_req_col["Service_Rank__c"].notnull()]
     ...: clients_req_col.isnull().sum()/len(clients_req_col)*100
Out[317]:
Id                              0.000000
Client__c                       0.000000
Date_of_Service_EntryNew__c     0.000000
Date_of_SeparationNew__c        0.000000
Service_Branch__c               0.000000
Service_Rank__c                 0.000000
Military_Spouse_Caregiver__c    0.000000
CreatedDate                     0.000000
Alumni_Survey_Completed__c      2.997378
Active_Color__c                 0.001449
EntryDate                       0.000000
LastDate                        0.000000
Service_Period                  0.000000
Joining_Period                  0.000000
dtype: float64
```

From the updated table we can see that only Alumni_Survey_Completed__c and Active_Color_c variables have missing values. But these columns are required only to answer second question (Likelyhood of the survey) So by removing these columns we create the final

data set required for the analysis of relation between the demographics and when the client has joined HireHerosUSA.

Aligning the Rank values in both data sources to merge

The Rank values have been adjusted in the external data source in order get them aligned with the values in HireHerosUSA datasets.

```
#replace 0num values with 0-num to correspond to Service_Rank__c in Client Services
military_data_req['Pay Grade'].replace(['O10', 'O9', 'O8', 'O7', 'O6', 'O5', 'O4', 'O3', 'O2', 'O1'],
                                       ['O-10','O-9','O-8','O-7','O-6','O-5','O-4','O-3','O-2','O-1'],
                                       inplace=True)
#view the replaced values
military_data_req['Pay Grade'].unique()

military_data_req['Pay Grade'].replace(['W5', 'W4', 'W3', 'W2', 'W1'],
                                       ['W-5','W-4', 'W-3', 'W-2','W-1'],
                                       inplace=True)

#view the replaced values
military_data_req['Pay Grade'].unique()

military_data_req['Pay Grade'].replace(['E9', 'E8', 'E7', 'E6','E5', 'E4', 'E3', 'E2', 'E1'],
                                       ['E-9', 'E-8','E-7','E-6','E-5','E-4','E-3','E-2','E-1'],
                                       inplace=True)
```

Removing Null values for data related to Question 2

```
In [325]: clients_req_col_Alumni.shape
Out[325]: (27380, 10)

In [326]: clients_req_col_Alumni.isnull().sum()
Out[326]:
Id                              0
Client__c                       0
Service_Branch__c               0
Service_Rank__c                 0
Military_Spouse_Caregiver__c    0
Alumni_Survey_Completed__c      5
Active_Color__c                 0
Service_Period                  0
Col                             0
Rank_Description                0
dtype: int64
```

From the table we can see that just 5 rows has na values. These rows have been eliminated from the dataset in order to get the final data set required for the analysis of likelihood of survey completion.

```
In [327]: clients_req_col_Alumni.dropna(inplace=True)
     ...: clients_req_col_Alumni.isnull().sum()
Out[327]:
Id                              0
Client__c                       0
Service_Branch__c               0
Service_Rank__c                 0
Military_Spouse_Caregiver__c    0
Alumni_Survey_Completed__c      0
Active_Color__c                 0
Service_Period                  0
Col                             0
Rank_Description                0
dtype: int64

In [328]:
```

Data Transformation

Creation of Calculated variables

From the StartDate and EndDate we derived the duration of clients service at armed forces in number of months.

```
#Calculation of Service period in number of months
clients_req_col['Service_Period'] = ((clients_req_col.LastDate - clients_req_col.EntryDate)/np.timedelta64(1, 'M'))
clients_req_col['Service_Period']
```

From the EndDate and Created Date we derived the Joining period (How early or late is he joining HireHerosUSA from his last day at armed force in months)

```
#calculating the gap between the joining date and retirement date.
clients_req_col['Joining_Period'] = ((clients_req_col.LastDate - clients_req_col.CreatedDate)/np.timedelta64(1, 'M'))
clients_req_col['Joining_Period']
```

Data Reduction

```
153 #dummy variable for Service_Rank__c
154 rank_data = pd.read_csv("D:\\MSIS 5633\\Clients_Military.csv")
155 rank_data.sort_values(by='Service_Rank__c')
156 rank_data['Service_Rank__c'] = rank_data['Service_Rank__c'].astype('category')
157
158 rank_dummy1 = pd.get_dummies(rank_data['Service_Rank__c'], prefix='rank')
159
160 rank_dummy1.head()
161
162 rank_dummy1.columns = ['rank1', 'rank2','rank3','rank4','rank5',
163                        'rank6','rank7','rank8','rank9','rank10',
164                        'rank11','rank12','rank13','rank14','rank15',
165                        'rank16','rank17','rank18','rank19','rank20',
166                        'rank21','rank22','rank23']
167
168 rank_data = rank_data.join(rank_dummy1)
169 #creating a new dataframe rank_data3 including dummy variables
170 rank_data3 = rank_data['rank1','rank2','rank3','rank4','rank5',
171                        'rank6','rank7','rank8','rank9','rank10',
172                        'rank11','rank12','rank13','rank14','rank15',
173                        'rank16','rank17','rank18','rank19','rank20',
174                        'rank21','rank22','rank23','cols1','cols2','cols3']
175 km = cls.KMeans(n_clusters=23).fit(rank_data3)
176 km.labels_
177
```

```
#dummy variable for Col
rank_data.sort_values(by='Col')
rank_data['Col'] = rank_data['Col'].astype('category')

rank_dummy2 = pd.get_dummies(rank_data['Col'], prefix='cols')

rank_dummy2.head()

rank_dummy2.columns = ['cols1','cols2','cols3']

rank_data = rank_data.join(rank_dummy2)

km = cls.KMeans(n_clusters=3).fit(rank_data3)
km.labels_
```

```
In [129]: km =
cls.KMeans(n_clusters=23).fit(rank_data2)

In [130]: km.labels_
Out[130]: array([ 2,  3,  0, ..., 22, 12, 22])

In [131]: km = cls.KMeans(n_clusters=3).fit(rank_data2)

In [132]: km.labels_
Out[132]: array([2, 0, 0, ..., 1, 1, 1])

In [133]: rank_data = pd.read_csv("D:\\MSIS 5633\
\Clients_Military.csv")
```

We have the Ranks classified into 23 categories from the HireHerosUSA data with various pay level codes and we also have high level classification these Ranks into three categories. As both the variables represents the pay level it is good to get rid of one of these two. To know which variable best explains the variation in data we should apply one of the Data Reduction Techniques. As we are looking at Categorical variables it is best to use Cluster analysis as data reduction technique in this case.

Above are the results of the K-means cluster analysis that has been performed on the data set with 23 clusters and 3 clusters based on the number of unique values in each of the Rank related variables. From the results we can see that the variable 'Col' (3 categories) best explains the classification of data in our data set. As a result we will eliminate the Service_Rank__c column from the data set.

Descriptive Statistics

The descriptive statistics below shows the mean shows the means, standard deviation,

Minimum and maximum values for all the variables

```
In [23]: rank_data['Joining_Period'].describe()
Out[23]:
count    69027.000000
mean       -24.026337
std         78.931099
min      -1325.234604
25%        -27.302409
50%          2.201277
75%          8.115156
max       1021.721185
Name: Joining_Period, dtype: float64
```

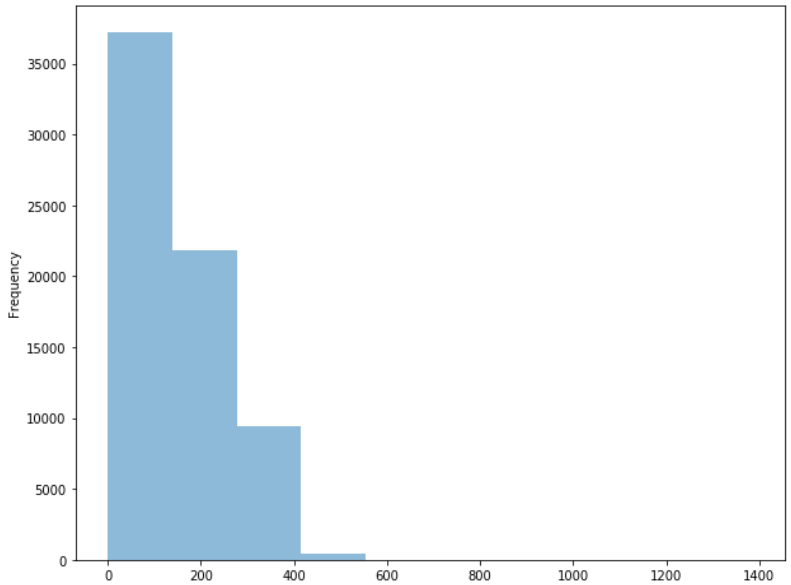
```
In [66]: alumni_data['Rank_Description'].describe()
Out[66]:
count        27375
unique          48
top       Sergeant
freq          3784
Name: Rank_Description, dtype: object

In [67]: alumni_data['Service_Period'].describe()

Out[67]:
count    27375.000000
mean       161.032053
std        103.643748
min          0.032855
25%         71.952196
50%        130.696729
75%        245.524549
max       1385.063348
Name: Service_Period, dtype: float64Out[68]:
count     27375
unique        5
top        Army
freq      14767
Name: Service_Branch__c, dtype: object
```

```
In [24]: rank_data['Service_Branch__c'].describe()
Out[24]:
count     69027
unique        5
top        Army
freq      37294
Name: Service_Branch__c, dtype: object
```

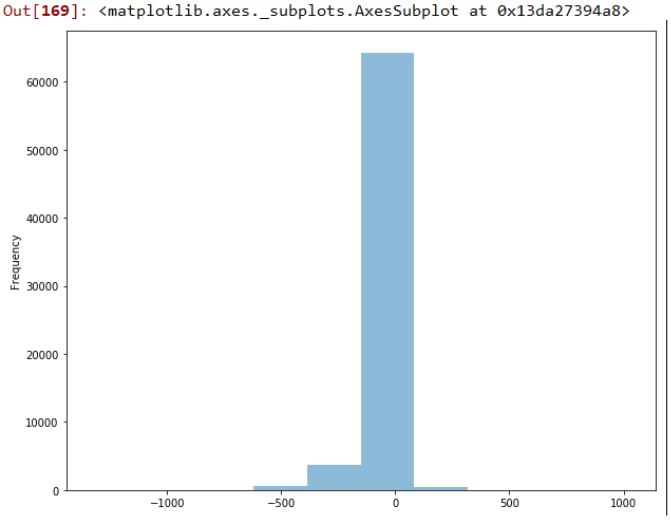
```
In [25]: rank_data['Service_Rank__c'].describe()
Out[25]:
count     69027
unique       23
top          E-4
freq      15301
Name: Service_Rank__c, dtype: object
```

```
In [26]: rank_data['Col'].describe()
Out[26]:
count                 69027
unique                    3
top       ENLISTED MEMBERS
freq                  59287
Name: Col, dtype: object
```

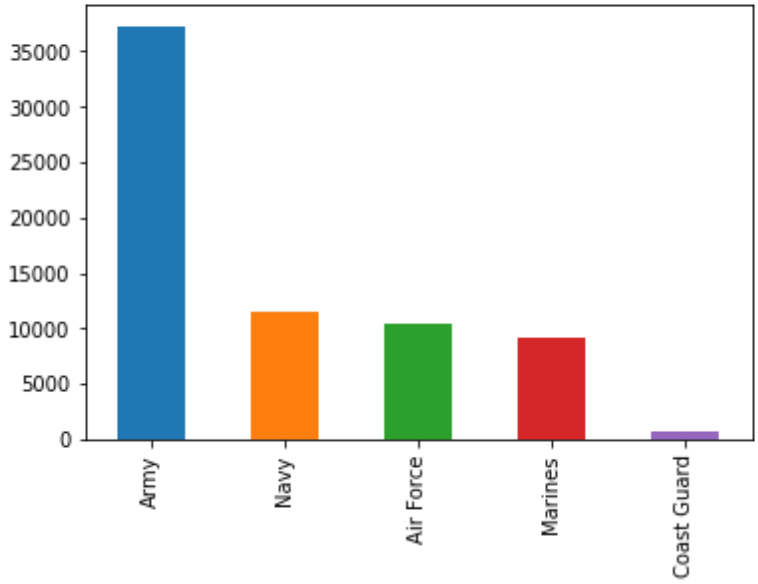
```
In [167]: rank_data['Service_Period'].plot.hist(alpha=0.5)
Out[167]: <matplotlib.axes._subplots.AxesSubplot at 0x13da25b6a20>
```



```
In [169]: rank_data['Joining_Period'].plot.hist(alpha=0.5)
Out[169]: <matplotlib.axes._subplots.AxesSubplot at 0x13da27394a8>
```

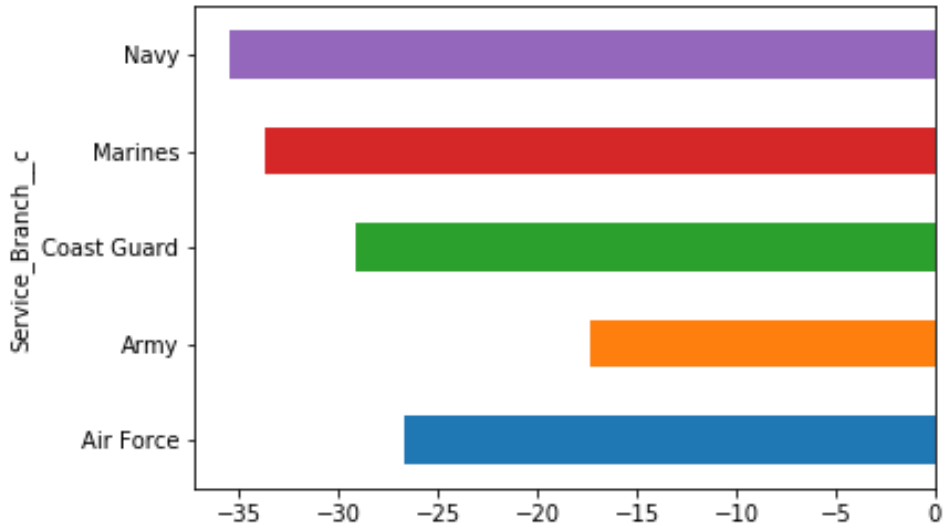



```
rank_data['Service_Branch__c'].value_counts().plot(kind='bar'
)
Out[26]: <matplotlib.axes._subplots.AxesSubplot at
0x25bf950ddd8>
```

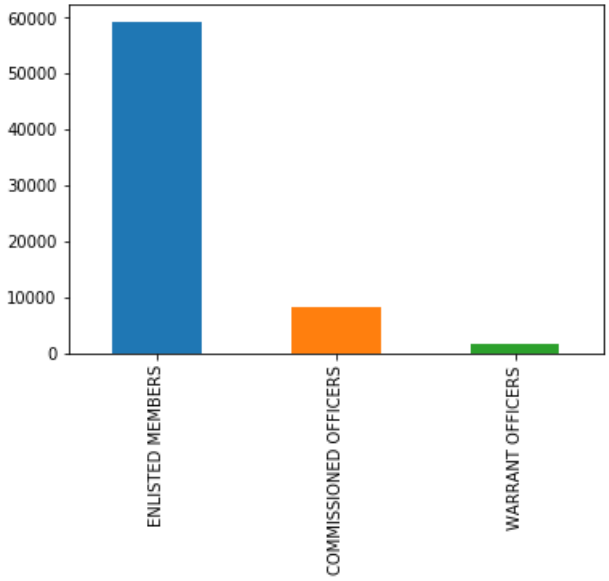


Graph shows the count of the members according to their branches.

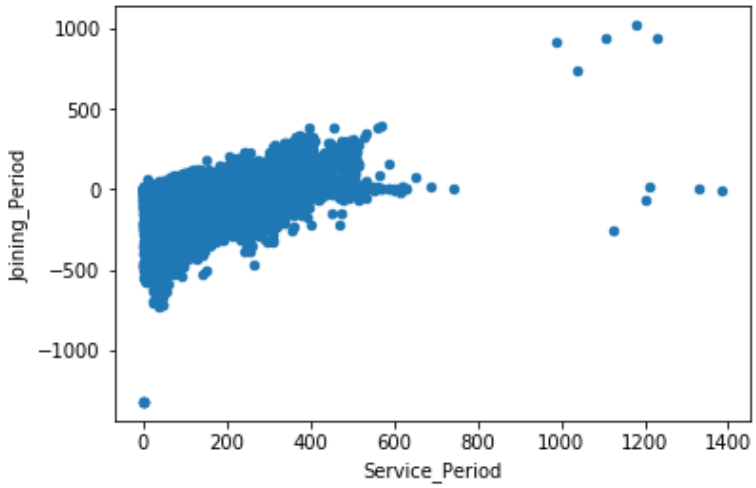
```
In [31]: rank_data.groupby('Service_Branch__c')
["Joining_Period"].mean().plot(kind='barh')
Out[31]: <matplotlib.axes._subplots.AxesSubplot at
0x25bf991c898>
```




```
In [27]: rank_data['Col'].value_counts().plot(kind='bar')
Out[27]: <matplotlib.axes._subplots.AxesSubplot at
0x23d415059e8>
```

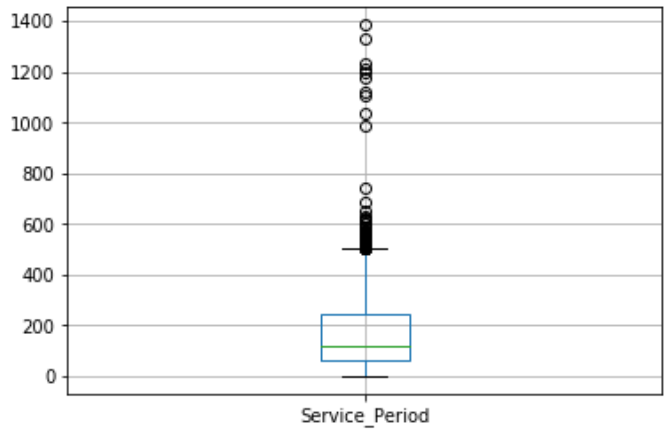


```
rank_data.plot.scatter(x='Service_Period',y='Joining_Period')
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x23d2f2082e8>
```

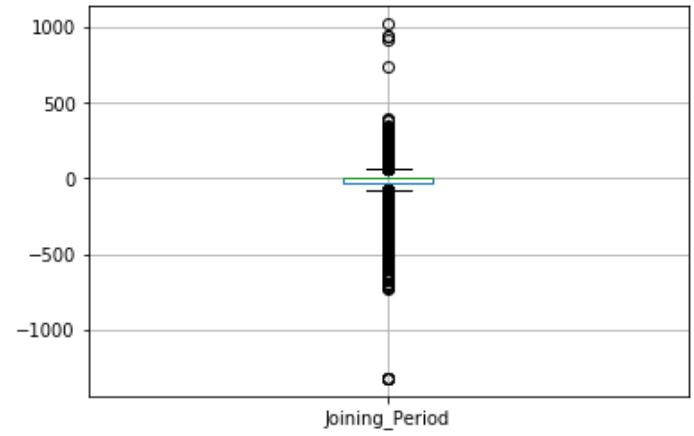


The above graph shows somewhat linear pattern in the initial ranges and we can clearly observe few outliers.

```
In [8]: rank_data.boxplot(column='Service_Period')
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x23d3db0f470>
```

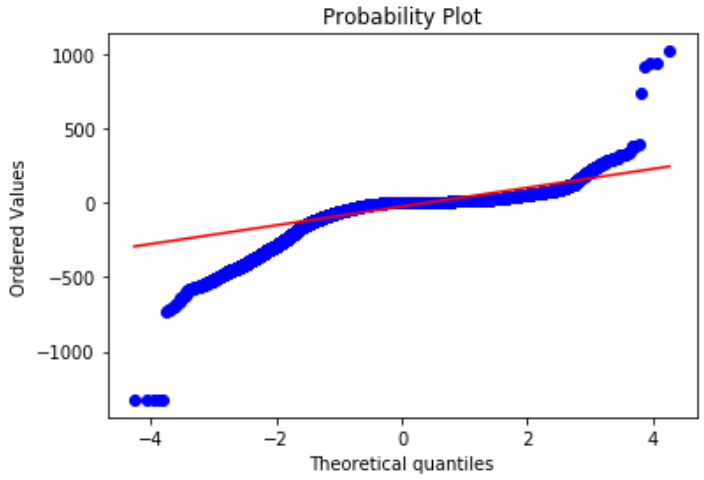


```
In [9]: rank_data.boxplot(column='Joining_Period')
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x23d3da3a198>
```



We can observe outliers in the above Box Plots.

```
In [13]: sts.probplot(rank_data.Joining_Period, dist="norm", plot=plt)
Out[13]:
((array([-4.26396309, -4.06154613, -3.95131544, ...,  3.95131544,
          4.06154613,  4.26396309]),
  array([-1325.23460441, -1323.32902113, -1323.06618206, ...,
           946.41779092,   946.58206534,  1021.72118524])),
 (63.211957580173795, -24.02633688527384, 0.8008062704057798))
```



The data is somewhat heavily tailed.

sts.shapiro(rank_data.Joining_Period)

Out[64]: (0.6414784789085388, 0.0)

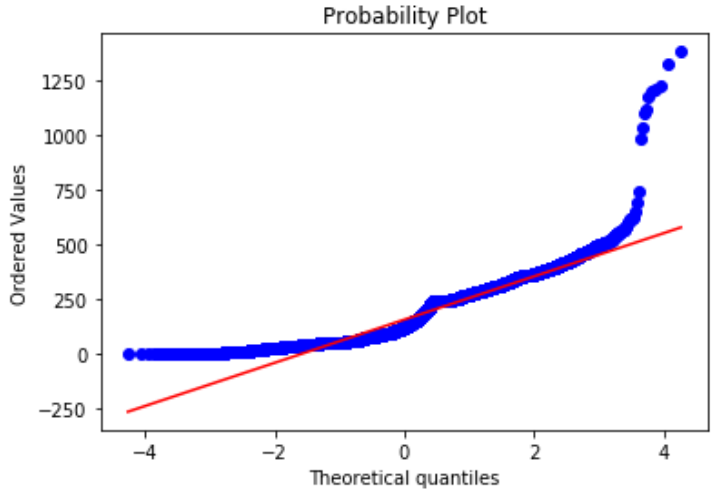
Note: p-value may not be accurate for N > 5000.

In the above Shapiro-Wilk test output,

first argument = value of test statistic and second argument = p-value. We find p-value is less

than 0.05. Hence we reject the null-hypothesis that radiation is normally distributed.

```
In [14]: sts.probplot(rank_data.Service_Period, dist="norm", plot=plt)
Out[14]:
((array([-4.26396309, -4.06154613, -3.95131544, ...,  3.95131544,
          4.06154613,  4.26396309]),
  array([3.28548841e-02, 3.28548841e-02, 3.28548841e-02, ...,
         1.23130249e+03, 1.33118134e+03, 1.38506335e+03])),
 (99.07128442557409, 156.38430051553848, 0.9547042011820948))
```



Data is observed in bow shape and by which we can say it is definitely not linear.

sts.shapiro(rank_data.Service_Period)

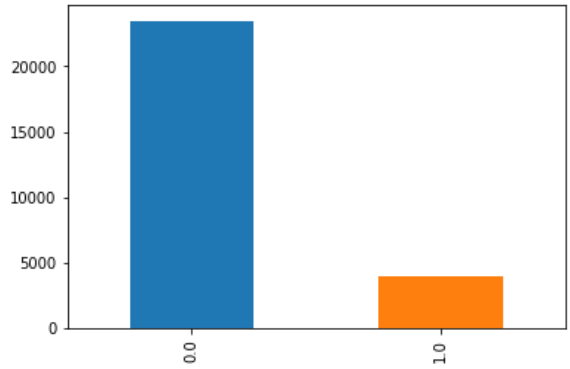
Out[65]: (0.9114835858345032, 0.0)

From the above Shapiro-Wilk test output, We find p-value is less

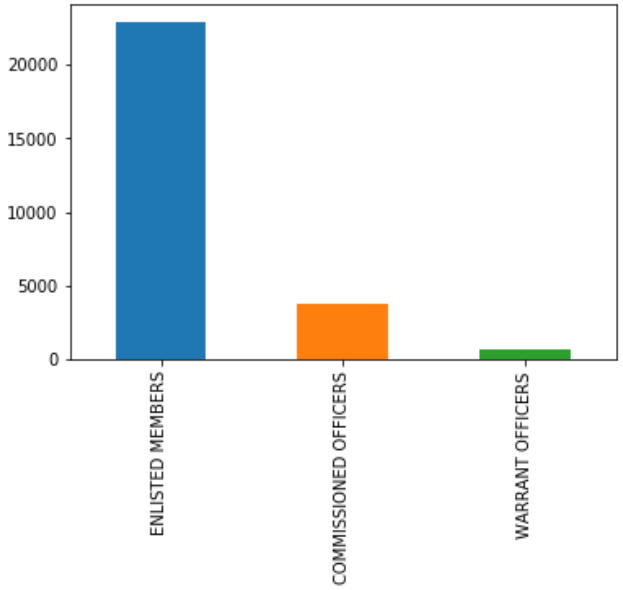
than 0.05. Hence we reject the null-hypothesis that radiation is normally distributed

In the below plot, we see that the ones who complete the survey are much less than the ones who complete.

```
In [52]:
alumni_data['Alumni_Survey_Completed__c'].value_counts().plot(kind='bar')
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x23d436b7fd0>
```

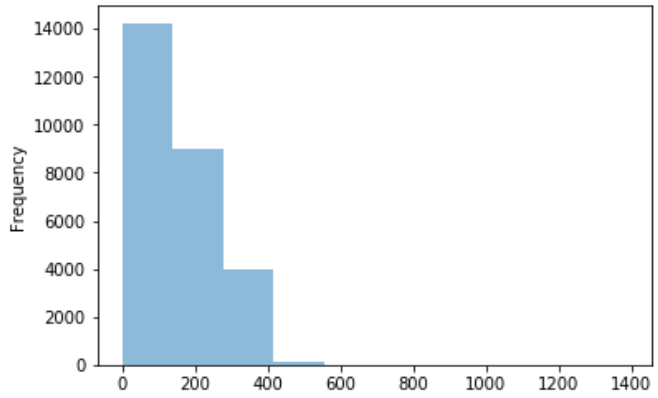



```
In [53]: alumni_data['Col'].value_counts().plot(kind='bar')
Out[53]: <matplotlib.axes._subplots.AxesSubplot at
0x23d436f06d8>
```

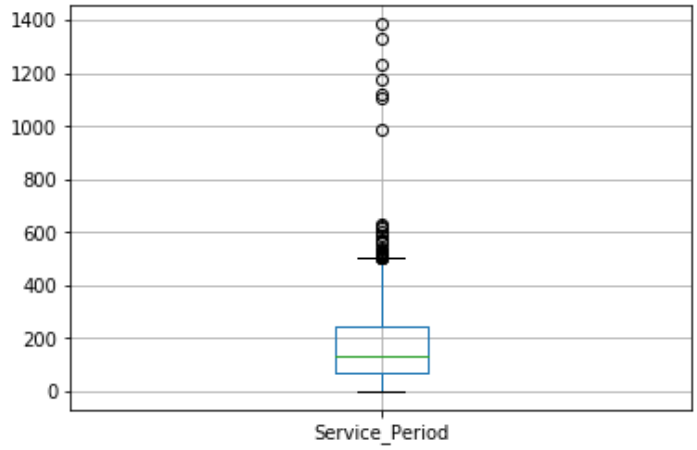


The following graph shows the member's categories and the count having Enlisted Members at the highest followed by Commissioned Officer and Warrant officers with the least value.

```
In [49]:
alumni_data['Service_Period'].plot.hist(alpha=0.5)
Out[49]: <matplotlib.axes._subplots.AxesSubplot at
0x23d435cc2b0>
```

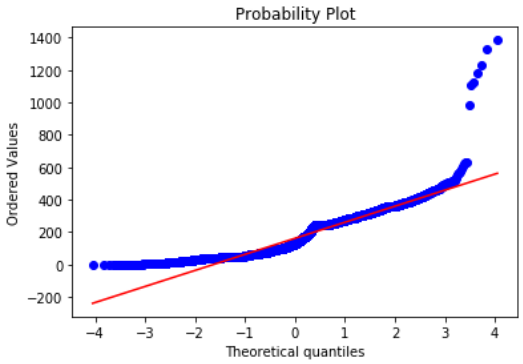



```
In [57]: alumni_data.boxplot(column='Service_Period')
Out[57]: <matplotlib.axes._subplots.AxesSubplot at
0x23d43a0f9e8>
```



From the boxplot we can clearly see outliers are at a large number.

```
In [59]: sts.probplot(alumni_data.Service_Period, dist="norm", plot=plt)
Out[59]:
((array([-4.05265224, -3.84022672, -3.7241439 , ...,  3.7241439 ,
          3.84022672,  4.05265224]),
  array([3.28548841e-02, 3.28548841e-02, 3.28548841e-02, ...,
         1.23130249e+03, 1.33118134e+03, 1.38506335e+03])),
 (99.04073104474331, 161.0320532211617, 0.9554683511723813))
```



The above plot show a bow shape pattern .Thus we can see that it is not linear.

sts.shapiro(alumni_data.Service_Period)

Out[62]: (0.91303551197052, 0.0)

Data Dictionary

Attribute Names	Description	Data types	Source
Id	Unique Salesforce Identifier (individual contact)	Object	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
Client__C	True / False (Indicates the account is that of a job seeker)	Integer	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
Date_of_Service_EntryNew__C	First day of client's military service	Object	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets

Date_of_SeparationNew__C	Last day of client's military service (actual or anticipated)	Object	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
Service_Branch__C	Branch of military service (Army, Navy, etc.)	Category	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
Service_Rank__C	Most recent pay grade of job seeking client (if military service / E-4, O-7, etc.)	Category	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
Military_Spouse_Caregiver__C	True / False (if true, indicates job seeking client is a spouse of veteran / servicemember and does not have military service)	Integer	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
CreatedDate	Date unique account was established	Datetime	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets
EntryDate	First day of client's military service	Datetime	Manual creation
LastDate	Last day of client's military service (actual or anticipated)	Datetime	Manual creation
Service_Period	Time in military service in months (difference between separation date and entry date in miltary)	Float	Manual creation
Joining_Period	Difference between military separation date and created date in Hireheroes	Float	Manual creation
Col	Administrative classification of ranks by paygrade and branch of military service	Category	https://militarybenefits.info/armed-forces-comparative-pay-grades-and-ranks/
Rank_Description	Listing of ranks by paygrade and branch of military service	Category	https://militarybenefits.info/armed-forces-comparative-pay-grades-and-ranks/

Alumni_Survey_Completed__c	True / False (if true, client responded to / completed alumni program survey)	Integer	https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets

Adjusting the Deliverable 1 Requirements

General Project Information

We have not made any changes in the title of our project or the team members.

Executive Summary

For this data challenge, we have analyzed the data related to the 'Hire Heroes USA: Clients Services program' to provide valuable insights into how the clients demographic profile influences when a particular client completes an alumni survey.

Project schedule and duration changes

We did not make any changes to our initial schedule. We have planned our schedule very well since the beginning.

Statement of Scope

For the deliverable 2, we will be concentrating only on the 2nd business question which is : how the factors like rank, branch, spouse status etc. affect the client's likelihood to complete a survey? . That is we are trying to understand whether the factors like the branch of military or the pay grades of the clients or their time in military service helps to determine whether an alumni (who has a job through the guidance of Hire Heroes) completed the alumni program survey or not. This will help in knowing the clients better and smoothen the process of surveys going forward by targeting specific areas concerned.

Modeling Techniques:

We are using the following two modelling techniques in our project.

Decision tree

The model Decision Tree is chosen as the first model for analysis. This model generally uses divide and conquer method to represent data. This algorithm helps in visualizing the decision making processes wherein an optimal solution is presented at the end of each node. The interpretations are easily explainable which is one of the factors giving Decision Tree an edge over other algorithms.

Logistic Regression

Logistic Regression was chosen as the second model for the analysis as the algorithm is apt for our case where the dependent variable is categorical. In our model, the target variable "Alumni_Survey_Completed__c" is a binary categorical variable which has values of '0' or '1' .

Model Assumption

Decision tree

Being a non parametric algorithm there are not any assumptions made on the distribution of data or errors. In other words the type of data decides the construction of the model.

Logistic Regression

Logistic Regression demands the variables to be independent of each other. The multicollinearity is assumed to be very little or not at all among the independent variables. The independent variables and log odds are assumed to be linear. Having a fairly large data set in one of the assumed conditions with the absence of any outliers in the continuous predictors. These assumptions have been met in the model.

Data Splitting and Subsampling

In order to estimate the likelihood of an Alumni Completing a Survey we need to run regression model with target variable as “Alumni_Survey_Completed__c”. Below is the distribution of the target variable.

```
In [326]: alumni_data.groupby("Alumni_Survey_Completed__c")["Id"].count()
Out[326]:
Alumni_Survey_Completed__c
0    21332
1     3594
Name: Id, dtype: int64
```

In order to continue with Logistic regression we need to ensure equal variance among the variables. One way to ensure equal variance is to do stratified sampling.

This has been achieved by splitting the data into two parts, one with Alumni_Survey_Completed__c variable = 0 and the other with Alumni_Survey_Completed__c =1.

```
In [327]: Survey_Yes = alumni_data[alumni_data["Alumni_Survey_Completed__c"]==1]

In [328]: Survey_No = alumni_data[alumni_data["Alumni_Survey_Completed__c"]==0]
```

And then the 0’s are randomly sampled to pick the same number of records as of 1’s. and the sampled 0’s are appended with the 1’s dataset.

```
In [329]: Survey_No_Sample = Survey_No.sample(n=3594)

In [330]: Net_Data=Survey_No_Sample.append(Survey_Yes,ignore_index=True)
```

The resultant dataset is now split into testing and training data sets with 20% and 80% partitions respectively.

```
In [331]: X1 = Net_Data.iloc[:,10:]

In [332]: X1 = X1.join(Net_Data['Service_Period'])

In [333]: X1 = X1.join(Net_Data['Col'])

In [334]: y1 = Net_Data.iloc[:,7:8]

In [335]: X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size = 0.2)
```

Comparison of variables between test and training data

Training Data: target variable

```
In [337]: y_train.describe()
Out[337]:
       Alumni_Survey_Completed__c
count                 5750.000000
mean                     0.500174
std                      0.500043
min                      0.000000
25%                      0.000000
50%                      1.000000
75%                      1.000000
max                      1.000000
```

Training data input variables

```
In [338]: X_train.describe()
Out[338]:
         Air Force         Army      ...       Service_Period          Col
count  5750.000000  5750.000000      ...          5750.000000  5750.000000
mean      0.161739     0.522087      ...           172.712326     0.356870
std       0.368243     0.499555      ...           103.929978     0.746966
min       0.000000     0.000000      ...             0.459968     0.000000
25%       0.000000     0.000000      ...            72.050761     0.000000
50%       0.000000     1.000000      ...           155.994990     0.000000
75%       0.000000     1.000000      ...           254.518573     0.000000
max       1.000000     1.000000      ...           628.021109     2.000000

[8 rows x 9 columns]
```

Testing Data: target variable

```
In [339]: y_test.describe()
Out[339]:
       Alumni_Survey_Completed__c
count                 1438.000000
mean                     0.499305
std                      0.500173
min                      0.000000
25%                      0.000000
50%                      0.000000
75%                      1.000000
max                      1.000000
```

describing the input variables of the testing data

```
In [340]: X_test.describe()
Out[340]:
         Air Force         Army     ...       Service_Period          Col
count  1438.000000  1438.000000     ...          1438.000000  1438.000000
mean      0.167594     0.534771     ...           173.961539     0.368567
std       0.373635     0.498963     ...           110.122270     0.752939
min       0.000000     0.000000     ...             2.529826     0.000000
25%       0.000000     0.000000     ...            72.017906     0.000000
50%       0.000000     1.000000     ...           153.563728     0.000000
75%       0.000000     1.000000     ...           256.572004     0.000000
max       1.000000     1.000000     ...          1331.181338     2.000000

[8 rows x 9 columns]
```

From the above results we can clearly see that, all the variables are exhibiting similar nature across both test and training datasets.

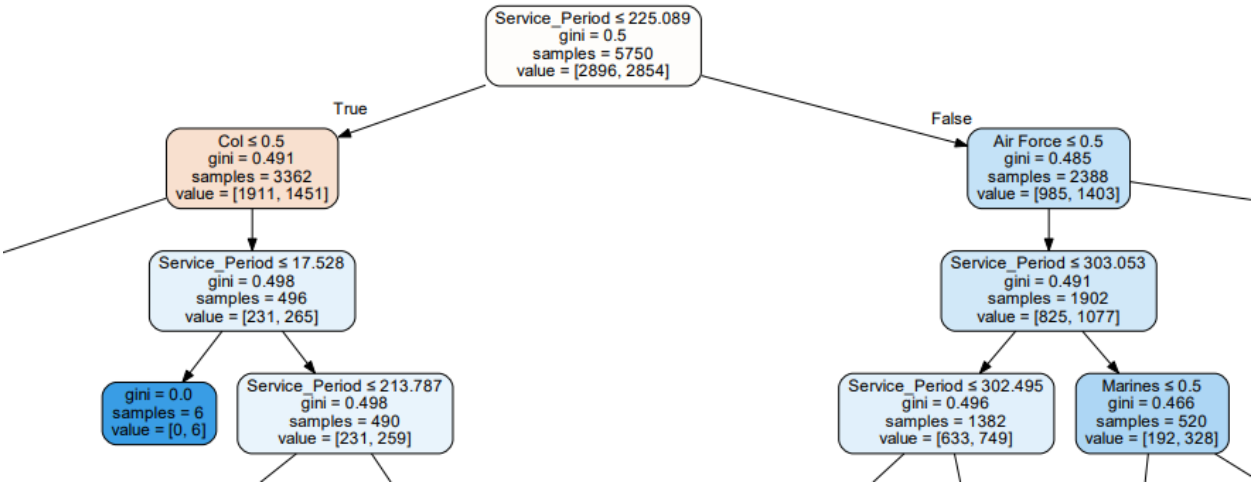
Model Building

Decision Tree Model

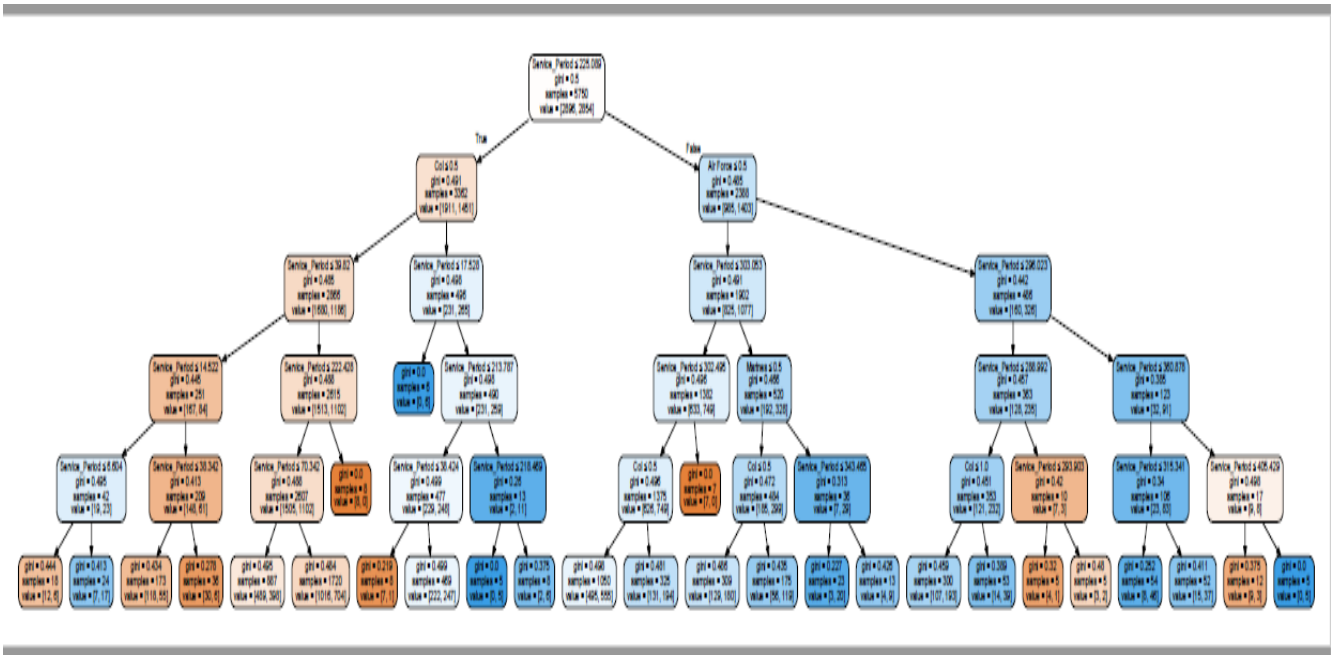
As the target variable is categorical variable we considered running Decision Tree Model. The model has been run with max_depth of 5 levels. Below are the results of the model.

```
In [349]: decisionTree.fit(X_train_s, y_train_s)
Out[349]:
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=100,
            splitter='best')
```

First level of decision tree

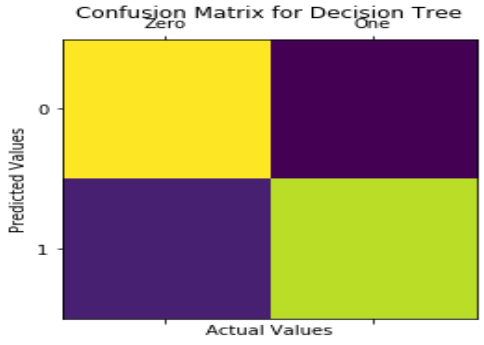


Based on the above tree, we see that the first criteria for the split is whether the service_period is less than or equal to 225 months or greater than this. If the service period is greater than the one specified, the fact whether the military category is Air force less than or equal to 0.5 is the next criteria for split. Depending upon the service period, the payscale of the military or veterans increase.



Classification Matrix:

```
In [363]: tree_prediction1 = decisionTree1.predict(X_test)

In [364]: cm1=confusion_matrix(y_test,tree_prediction1)

In [365]: plt.matshow(cm1)
     ...: plt.title("Confusion Matrix for Decision Tree")
     ...: plt.xlabel("Actual Values")
     ...: plt.ylabel("Predicted Values")
     ...: plt.xticks([0,1],['Zero','One'])
Out[365]:
([<matplotlib.axis.XTick at 0x1c2c840f0f0>,
  <matplotlib.axis.XTick at 0x1c2c840f898>],
 <a list of 2 Text xticklabel objects>)
```




```
In [366]: cm1
Out[366]:
array([[418, 302],
       [312, 406]], dtype=int64)
```

Results and Interpretation

From the above confusion matrix, we can say that the accuracy rate of the model is (418+406) / (418+302+312+406) = 824 / 1438 = 57. 3%

Hence the inaccuracy predicted by the model is = (1 - 57.3)= 42.7%

As such this is comparatively a good model.

Logistic Regression Model

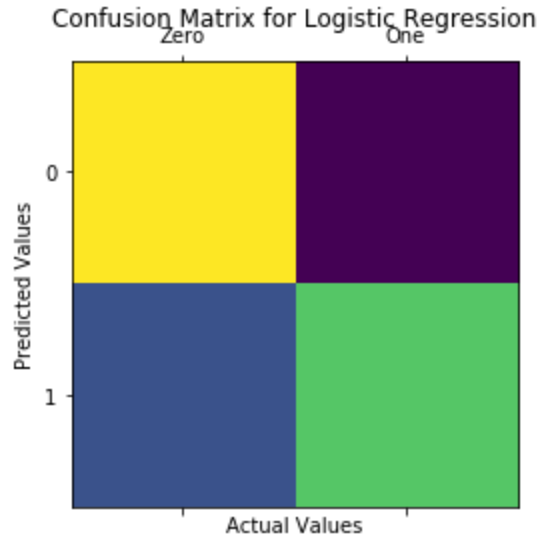
As we are trying to look at the likelihood of an event logistic regression is by default a model to check as it uses the likelihood technique to optimize the model. Below are results of this regression model.

```
In [367]: regressor1 = LogisticRegression()
     ...: regressor1.fit(X_train, y_train)
     ...: predicted_Survey_Compl1 = regressor1.predict(X_test)
     ...: regressor1.coef_
     ...:
C:\Users\anoop\Anaconda3\lib\site-packages\sklearn\linear_model
\logistic.py:433: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\anoop\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
761: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  y = column_or_1d(y, warn=True)
Out[367]:
array([[ 0.19527768, -0.18913868, -0.17945826, -0.01930606,
-0.05911714,
        -0.09839104, -0.15335142,  0.00239793,  0.18646823]])
```

Confusion Matrix:

```
In [368]: confusion_matrix(y_test,predicted_Survey_Compl1)
Out[368]:
array([[442, 278],
       [319, 399]], dtype=int64)
```

```
In [370]: plt.matshow(cm2)
     ...: plt.title("Confusion Matrix for Logistic Regression")
     ...: plt.xlabel("Actual Values")
     ...: plt.ylabel("Predicted Values")
     ...: plt.xticks([0,1],['Zero','One'])
Out[370]:
([<matplotlib.axis.XTick at 0x1c32bd4e160>,
  <matplotlib.axis.XTick at 0x1c32bb53c18>],
 <a list of 2 Text xticklabel objects>)
```



Results and Interpretation

The coefficients of the model suggests the below:

- Log Odds ratio of an Alumni completing the survey decreases by 0.179 when they belongs to Air Force. Which mean that Air Force veterans are less likely to fill the Alumni Survey

- Log Odds ratio of an Alumni completing the survey decreases by 0.189 when they belongs to Army. Which mean that Army veterans are less likely to fill the Alumni Survey.

- Log Odds ratio of an Alumni completing the survey decreases by 0.019 when they belongs to Coast Guard. Which mean that Coast Guard veterans are less likely to fill the Alumni Survey.

- Log Odds ratio of an Alumni completing the survey decreases by 0.059 when they belongs to Marines. Which mean that Marines veterans are less likely to fill the Alumni Survey.

- Log Odds ratio of an Alumni completing the survey decreases by 0.098 when they belongs to Navy. Which mean that Navy veterans are less likely to fill the Alumni Survey.

- Log Odds ratio of an Alumni completing the survey decreases by 0.015 when the veteran is Male.

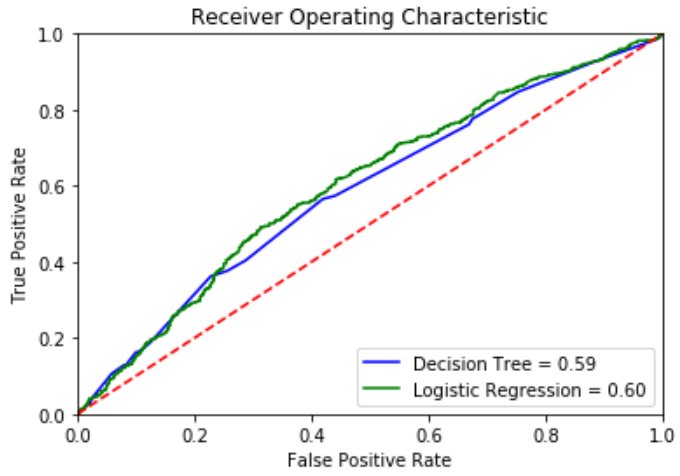
- Log Odds ratio of an Alumni completing the survey increase by 0.002 when the veteran's service period increases by a month.

- Log Odds ratio of an Alumni completing the survey increase by 0.186 when the veteran's Pay Rank increases by 1.

Model Assessments

ROC Curve:

```
In [377]: plt.title('Receiver Operating Characteristic')
     ...: plt.plot(fpr, tpr, 'b', label = 'Decision Tree = %0.2f' %
roc_auc)
     ...: plt.plot(fpr1, tpr1, 'g', label = 'Logistic Regression =
%0.2f' % roc_auc1)
     ...: plt.legend(loc = 'lower right')
     ...: plt.plot([0, 1], [0, 1],'r--')
     ...: plt.xlim([0, 1])
     ...: plt.ylim([0, 1])
     ...: plt.ylabel('True Positive Rate')
     ...: plt.xlabel('False Positive Rate')
     ...: plt.show()
     ...:
```



From the ROC curve both Decision Tree and Logistic Regression models are almost equally good. they have values 0.59 and 0.60. Which indicates that the Logistic regression is slightly having edge over the Decision Tree model. The same is evident with the Green curve has more area covered under the curve.

References

https://www.teradatauniversitynetwork.com/Community/Student-Competitions/2019/2019-Data-Challenge/Data-Challenge-Datasets

https://www.hireheroesusa.org/

https://www.federalpay.org/military/army/ranks