Why Juniors Quit

Import Packages

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import plotly as px

import warnings
warnings.filterwarnings('ignore')
warnings.warn('DelftStack')
warnings.warn('Do not show this message')
```

Import Dataset

In [2]:
```
jrquit = pd.read_excel(r"C:\Users\LinaNguyen\Desktop\JREMP.xlsx", index_col = 0)
```

In [3]:
```
jrquit.head()
```

Out[3]:

EmployeeNumber	DateOfBirth	State	RehireDate	StartDate	Gender	JobTitle	StartReason	EndDate	Terminated	RaceDescription	Certific
101121	1991-03-06	CA	NaN	2017-01-17	Male	Junior Analyst	New Hire	2017-08-18	True	NaN	
101131	1981-02-09	CA	NaN	2017-03-27	Male	Junior Analyst	New Hire	2018-05-31	True	NaN	
101142	1985-11-10	SC	NaN	2019-01-01	Male	Analyst	Reorganization	2019-06-19	True	NaN	
101146	1981-03-02	CA	NaN	2017-06-19	Female	Coordinator, Human Resources	New Hire	2018-06-14	True	NaN	
101164	1992-12-25	CA	NaN	2019-09-01	Female	Analyst	Promotion	2020-02-14	True	Hispanic	

Data Preprocessing

In [4]:
```
jrquit.isna().sum()
```

Out[4]:
```
DateOfBirth                0
State                      0
RehireDate                57
StartDate                  0
Gender                     1
JobTitle                   0
StartReason                0
EndDate                   29
Terminated                 0
RaceDescription            5
Certification             53
Masters                   52
Other                     42
Undergraduate             14
TerminationReason         34
HourlyRate                 0
VeteranStatusDescription   2
dtype: int64
```

In [5]:
```
#fill na values in Certification, Masters, Other, Undergraduate with 0, else = 1
jrquit[['Masters', 'Certification', 'Other', 'Undergraduate']] = jrquit[['Masters', 'Certification', 'Other', 'Ur
```

In [6]:
```
#fill NA Veteran Status Description, and Race with DNP - Did Not Provide
#Terminiation reason that isn't provided will be replaced with na - not applicable
jrquit['VeteranStatusDescription'] = jrquit['VeteranStatusDescription'].fillna('nanvet')
jrquit['RaceDescription'] = jrquit['RaceDescription'].fillna('nanracedescription')
jrquit['Gender'] = jrquit['Gender'].fillna('nangender')
```

```
jrquit['TerminationReason'] = jrquit['TerminationReason'].fillna('nanterminationreason')
```

In [7]:
```
jrquit.isna().sum()
```

Out[7]:
```
DateOfBirth                 0
State                       0
RehireDate                 57
StartDate                   0
Gender                      0
JobTitle                    0
StartReason                 0
EndDate                    29
Terminated                  0
RaceDescription             0
Certification               0
Masters                     0
Other                       0
Undergraduate               0
TerminationReason           0
HourlyRate                  0
VeteranStatusDescription    0
dtype: int64
```

In [8]:
```
#drop Date Columns
jrquit = jrquit.drop(['EndDate'], axis = 1)
```

In [9]:
```
jrquit['Terminated'].value_counts()
percent = jrquit['Terminated'].value_counts('True')
percent
```

Out[9]:
```
False    0.508772
True     0.491228
Name: Terminated, dtype: float64
```

In [10]:
```
jrquit['TerminationReason'].value_counts()
```

Out[10]:
```
nanterminationreason                  34
Resignation - Career Opportunity      11
Layoff                                 3
Resignation                            2
Resignation - Convert to Government    2
Resignation - Personal reasons         2
Resignation - Relocation               2
Resignation - Management               1
Name: TerminationReason, dtype: int64
```

In [11]:
```
jrquit['Terminated'] = np.where((jrquit.Terminated == 'True'), 1, jrquit.Terminated)
jrquit['Terminated'] = np.where((jrquit.Terminated == 'False'), 0, jrquit.Terminated)
jrquit['JobTitle'] = np.where((jrquit.JobTitle == 'Junior Analyst'), "Analyst", jrquit.JobTitle)
```

In [12]:
```
for col in jrquit.columns:
    print('-' * 40 + col + '-' * 40 , end='-')
    display(jrquit[col].value_counts().head(10))
```

```
----------------------------------------DateOfBirth----------------------------------------
1991-03-06    1
1993-09-14    1
1997-12-23    1
1980-08-23    1
1994-02-23    1
1997-03-19    1
1993-12-02    1
1997-10-27    1
1995-12-15    1
1994-11-22    1
Name: DateOfBirth, dtype: int64
----------------------------------------State----------------------------------------
CA    34
SC    16
VA     7
Name: State, dtype: int64

----------------------------------------RehireDate----------------------------------------

Series([], Name: RehireDate, dtype: int64)
```

```
---------------------------------------StartDate---------------------------------------
2019-01-01    5
2022-03-16    3
2022-03-01    3
2022-06-01    2
2022-01-18    2
2021-03-16    2
2021-05-17    2
2017-01-17    1
2021-06-21    1
2020-08-17    1
Name: StartDate, dtype: int64
---------------------------------------Gender---------------------------------------
Female       31
Male         25
nangender     1
Name: Gender, dtype: int64
-------------------------------------JobTitle-------------------------------------
Analyst                           53
Coordinator, Human Resources       1
Accountant                         1
Administrator, Human Resources     1
Junior Recruiter                   1
Name: JobTitle, dtype: int64
-------------------------------------StartReason-------------------------------------

New Hire          51
Reorganization     5
Promotion          1
Name: StartReason, dtype: int64
-------------------------------------Terminated-------------------------------------
0    29
1    28
Name: Terminated, dtype: int64
-------------------------------------RaceDescription-------------------------------------
White                                      26
Hispanic                                   11
Asian                                       7
Black                                       6
nanracedescription                          5
Prefer Not to Answer                        1
Native Hawaiian or Other Pacific Islander     1
Name: RaceDescription, dtype: int64
-------------------------------------Certification-------------------------------------
0    53
1     4
Name: Certification, dtype: int64
-------------------------------------Masters-------------------------------------
0    52
1     5
Name: Masters, dtype: int64

-------------------------------------Other-------------------------------------
0    42
1    15
Name: Other, dtype: int64
-------------------------------------Undergraduate-------------------------------------
1    43
0    14
Name: Undergraduate, dtype: int64
-------------------------------------TerminationReason-------------------------------------
nanterminationreason                  34
Resignation - Career Opportunity      11
Layoff                                 3
Resignation                            2
Resignation - Convert to Government    2
Resignation - Personal reasons         2
Resignation - Relocation               2
Resignation - Management               1
Name: TerminationReason, dtype: int64

-------------------------------------HourlyRate-------------------------------------
26.4423    8
25.0000    5
28.8462    4
27.8846    3
24.0385    3
24.5192    2
26.0000    2
26.4400    2
28.1250    1
21.5600    1
Name: HourlyRate, dtype: int64
-------------------------------------VeteranStatusDescription-------------------------------------
Not a protected veteran    42
Protected Veteran          12
nanvet                      2
Declined to Answer          1
Name: VeteranStatusDescription, dtype: int64
```

In [13]:

```
for col in jrquit.columns:
    print('-' * 40 + col + '-' * 40 , end='-')
    display(jrquit[col].value_counts('True').head(10))
```

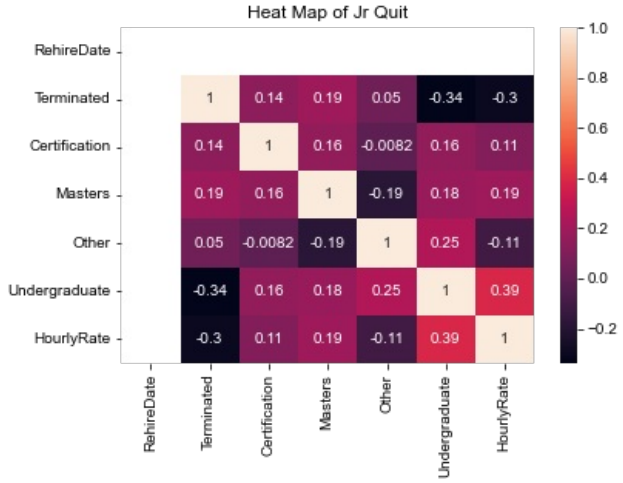
```
----------------------------------------DateOfBirth----------------------------------------
1991-03-06    0.017544
1993-09-14    0.017544
1997-12-23    0.017544
1980-08-23    0.017544
1994-02-23    0.017544
1997-03-19    0.017544
1993-12-02    0.017544
1997-10-27    0.017544
1995-12-15    0.017544
1994-11-22    0.017544
Name: DateOfBirth, dtype: float64


----------------------------------------State----------------------------------------
CA    0.596491
SC    0.280702
VA    0.122807
Name: State, dtype: float64

----------------------------------------RehireDate----------------------------------------

Series([], Name: RehireDate, dtype: float64)


----------------------------------------StartDate----------------------------------------
2019-01-01    0.087719
2022-03-16    0.052632
2022-03-01    0.052632
2022-06-01    0.035088
2022-01-18    0.035088
```

```
2021-03-16    0.035088
2021-05-17    0.035088
2017-01-17    0.017544
2021-06-21    0.017544
2020-08-17    0.017544
Name: StartDate, dtype: float64


----------------------------------------Gender----------------------------------------
Female       0.543860
Male         0.438596
nangender    0.017544
Name: Gender, dtype: float64
----------------------------------------JobTitle----------------------------------------


Analyst                           0.929825
Coordinator, Human Resources      0.017544
Accountant                        0.017544
Administrator, Human Resources    0.017544
Junior Recruiter                  0.017544
Name: JobTitle, dtype: float64
----------------------------------------StartReason----------------------------------------


New Hire          0.894737
Reorganization    0.087719
Promotion         0.017544
Name: StartReason, dtype: float64
----------------------------------------Terminated----------------------------------------

0    0.508772
1    0.491228
Name: Terminated, dtype: float64
----------------------------------------RaceDescription----------------------------------------
White                                       0.456140
Hispanic                                    0.192982
Asian                                       0.122807
Black                                       0.105263
nanracedescription                          0.087719
Prefer Not to Answer                        0.017544
Native Hawaiian or Other Pacific Islander    0.017544
Name: RaceDescription, dtype: float64
----------------------------------------Certification----------------------------------------

0    0.929825
1    0.070175
Name: Certification, dtype: float64

----------------------------------------Masters----------------------------------------
0    0.912281
1    0.087719
Name: Masters, dtype: float64
----------------------------------------Other----------------------------------------

0    0.736842
1    0.263158
Name: Other, dtype: float64

----------------------------------------Undergraduate----------------------------------------
1    0.754386
0    0.245614
Name: Undergraduate, dtype: float64
----------------------------------------TerminationReason----------------------------------------
nanterminationreason                  0.596491
Resignation - Career Opportunity      0.192982
Layoff                                0.052632
Resignation                           0.035088
Resignation - Convert to Government    0.035088
Resignation - Personal reasons        0.035088
Resignation - Relocation              0.035088
Resignation - Management              0.017544
Name: TerminationReason, dtype: float64

----------------------------------------HourlyRate----------------------------------------


26.4423    0.140351
25.0000    0.087719
28.8462    0.070175
27.8846    0.052632
24.0385    0.052632
24.5192    0.035088
26.0000    0.035088
26.4400    0.035088
28.1250    0.017544
21.5600    0.017544
Name: HourlyRate, dtype: float64

----------------------------------------VeteranStatusDescription----------------------------------------
Not a protected veteran    0.736842
```

```
Protected Veteran          0.210526
nanvet                     0.035088
Declined to Answer         0.017544
Name: VeteranStatusDescription, dtype: float64
```

In [14]:

```
sns.heatmap(jrquit.corr(), annot = True)
sns.set(rc={"figure.figsize":(20, 10)})
plt.title('Heat Map of Jr Quit')
plt.show()
```



Data Splitting

In [15]:

```
#view unique values to determine which columns to include during OneHotEncoding
jrquit.nunique()
```

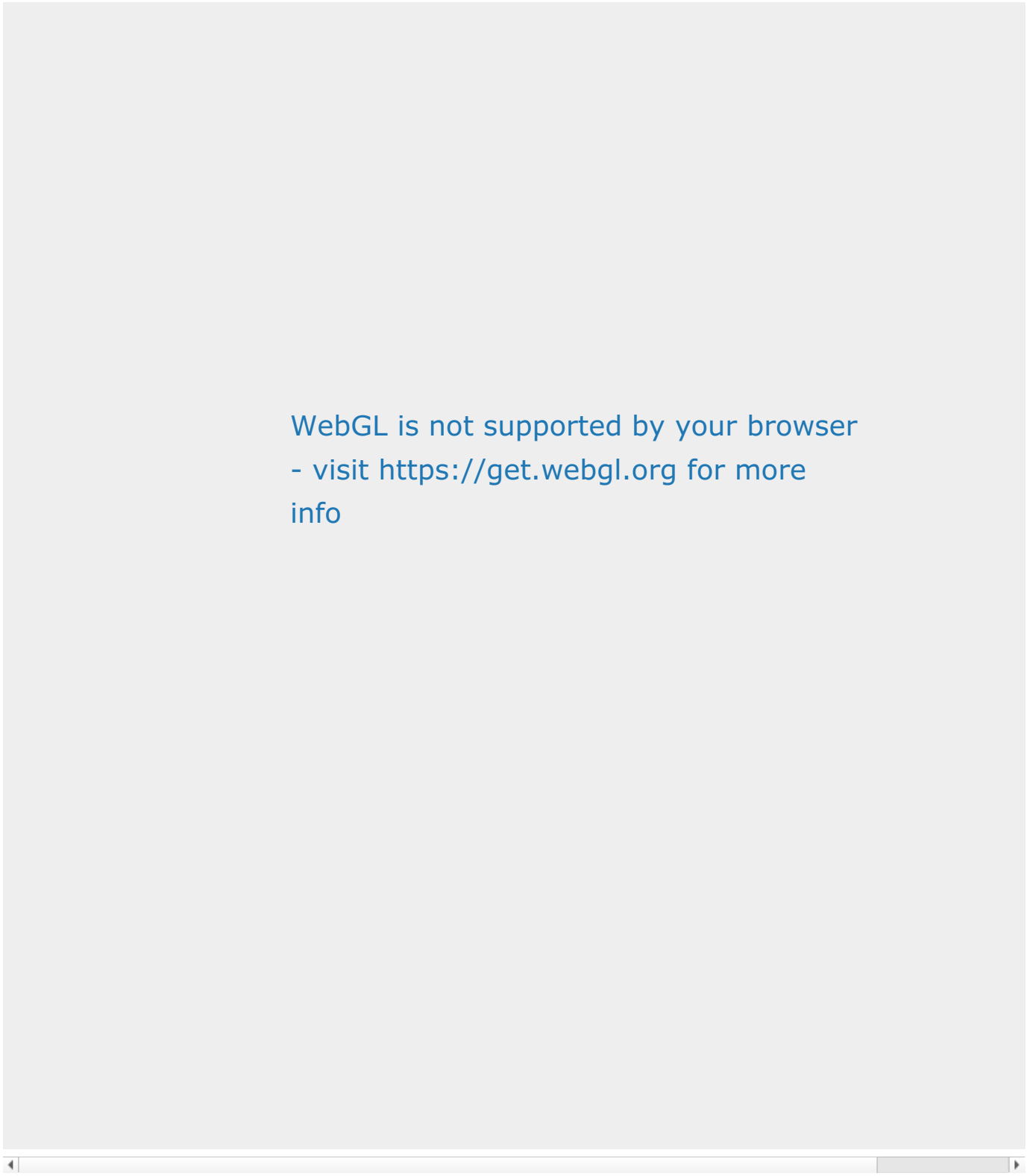
Out[15]:

```
DateOfBirth                57
State                       3
RehireDate                  0
StartDate                  45
Gender                      3
JobTitle                    5
StartReason                 3
Terminated                  2
RaceDescription             7
Certification               2
```

```
Masters                       2
Other                         2
Undergraduate                 2
TerminationReason             8
HourlyRate                   36
VeteranStatusDescription      4
dtype: int64
```

In [16]:

```
#checking for multicollinearity
import plotly.express as px
fig = px.scatter_matrix(jrquit, width = 1000, height = 1000)
fig.show()
```



Binning Analyst salary into low, average, and high based off of the median salary +/-3 per state. We are assuming junior analyst and analyst are the same. We are also assuming that HR salary is average as it¶

In [17]:

In [17]:
```
#Replace Junior Analyst with Analyst
jrquit['JobTitle'] = jrquit['JobTitle'].replace({'Junior Analyst': 'Analyst'}, regex = True)
jrquit['JobTitle'].value_counts()
```

Out[17]:
```
Analyst                           53
Coordinator, Human Resources       1
Accountant                         1
Administrator, Human Resources     1
Junior Recruiter                   1
Name: JobTitle, dtype: int64
```

In [18]:
```
#break data into hourly rate and job title
salary = jrquit[['HourlyRate', 'JobTitle', 'State']]
salary.head()
```

Out[18]:

EmployeeNumber	HourlyRate	JobTitle	State
101121	25.0000	Analyst	CA
101131	23.0769	Analyst	CA
101142	7.5175	Analyst	SC
101146	21.6400	Coordinator, Human Resources	CA
101164	19.8600	Analyst	CA

In [19]:
```
# view median salary
group1 = salary.groupby(['JobTitle', 'State'])['HourlyRate'].mean()
group1.to_frame()
```

Out[19]:

		HourlyRate
JobTitle	**State**	
Accountant	**CA**	34.375000
Administrator, Human Resources	**CA**	23.000000
Analyst	**CA**	24.463867
	SC	23.107525
	VA	27.472529
Coordinator, Human Resources	**CA**	21.640000
Junior Recruiter	**CA**	24.000000

In [20]:
```
#salary is all categorized together until we figure out how to do it separately. HR is categorized as low
salary['pay_cat'] = pd.cut(salary['HourlyRate'], bins = [0, 22, 28, 35], include_lowest = True, labels = ['low',
salary.head()
```

Out[20]:

EmployeeNumber	HourlyRate	JobTitle	State	pay_cat
101121	25.0000	Analyst	CA	average
101131	23.0769	Analyst	CA	average
101142	7.5175	Analyst	SC	low
101146	21.6400	Coordinator, Human Resources	CA	low
101164	19.8600	Analyst	CA	low

In [21]:
```
#group by job title per state
groups = salary.groupby(['JobTitle', 'State'])
for group_key, group_value in groups:
    group = groups.get_group(group_key)
    print(group)
    print('')
```

```
                HourlyRate    JobTitle State pay_cat
EmployeeNumber
101311              34.375  Accountant    CA    high

                HourlyRate                        JobTitle State  pay_cat
```

```
EmployeeNumber
101427               23.0  Administrator, Human Resources    CA  average
```

```
               HourlyRate JobTitle State  pay_cat
EmployeeNumber
101121            25.0000  Analyst    CA  average
101131            23.0769  Analyst    CA  average
101164            19.8600  Analyst    CA      low
101188            23.3173  Analyst    CA  average
101205            28.8462  Analyst    CA     high
101217            12.0193  Analyst    CA      low
101223            21.3675  Analyst    CA      low
101254            24.0385  Analyst    CA  average
101261            25.0000  Analyst    CA  average
101262            27.8846  Analyst    CA  average
101265            26.0000  Analyst    CA  average
101277            28.2212  Analyst    CA     high
101280            23.0800  Analyst    CA  average
101290            26.4423  Analyst    CA  average
101302            26.0000  Analyst    CA  average
101320            25.0000  Analyst    CA  average
101334            21.5600  Analyst    CA      low
101358            10.4022  Analyst    CA      low
101362            28.0288  Analyst    CA     high
101375            28.8462  Analyst    CA     high
101376            28.1250  Analyst    CA     high
101382            28.1300  Analyst    CA     high
101395            25.0000  Analyst    CA  average
101397            25.0000  Analyst    CA  average
101399            26.4400  Analyst    CA  average
101414            22.0000  Analyst    CA      low
101415            26.4400  Analyst    CA  average
101421            22.9100  Analyst    CA  average
101455            28.0000  Analyst    CA  average
101464            27.8800  Analyst    CA  average
```

```
               HourlyRate JobTitle State  pay_cat
EmployeeNumber
101142             7.5175  Analyst    SC      low
101167            21.2500  Analyst    SC      low
101174            20.8654  Analyst    SC      low
101184            24.0385  Analyst    SC  average
101193            25.2404  Analyst    SC  average
101204            26.9712  Analyst    SC  average
101207            20.1923  Analyst    SC      low
101316            24.0385  Analyst    SC  average
101318            24.5192  Analyst    SC  average
101325            26.4423  Analyst    SC  average
101344            24.5192  Analyst    SC  average
101363            26.4423  Analyst    SC  average
101380            18.3567  Analyst    SC      low
101404            26.4423  Analyst    SC  average
101430            26.4423  Analyst    SC  average
101431            26.4423  Analyst    SC  average
```

```
               HourlyRate JobTitle State  pay_cat
EmployeeNumber
101240            25.9615  Analyst    VA  average
101339            27.8846  Analyst    VA  average
101425            26.4423  Analyst    VA  average
101426            26.4423  Analyst    VA  average
101447            27.8846  Analyst    VA  average
101458            28.8462  Analyst    VA     high
101459            28.8462  Analyst    VA     high
```

```
               HourlyRate                      JobTitle State pay_cat
EmployeeNumber
101146              21.64  Coordinator, Human Resources    CA     low
```

```
               HourlyRate          JobTitle State  pay_cat
EmployeeNumber
101434               24.0  Junior Recruiter    CA  average
```

In [22]:
```
#check if imbalance exists
jrquit['Terminated'].value_counts()
#class imbalance does not exist
```

Out[22]:
```
0    29
1    28
Name: Terminated, dtype: int64
```

In [23]:
```
jrquit.columns
```

Out[23]:
```
Index(['DateOfBirth', 'State', 'RehireDate', 'StartDate', 'Gender', 'JobTitle',
       'StartReason', 'Terminated', 'RaceDescription', 'Certification',
       'Masters', 'Other', 'Undergraduate', 'TerminationReason', 'HourlyRate',
       'VeteranStatusDescription'],
      dtype='object')
```

In [24]:
```
classcol = jrquit[['JobTitle','State', 'Gender','StartReason','VeteranStatusDescription', 'RaceDescription', 'Ter
       'Certification', 'Masters', 'Other', 'Undergraduate',
       'TerminationReason']]
classcol.describe()
```

Out[24]:

	Terminated	Certification	Masters	Other	Undergraduate
count	57.000000	57.000000	57.000000	57.000000	57.000000
mean	0.491228	0.070175	0.087719	0.263158	0.754386
std	0.504367	0.257713	0.285401	0.444262	0.434277
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

In [25]:
```
salarycol = salary[['pay_cat']]
```

In [26]:
```
finalclean = pd.concat([classcol, salarycol], axis = 1)
finalclean.head()
```

Out[26]:

EmployeeNumber	JobTitle	State	Gender	StartReason	VeteranStatusDescription	RaceDescription	Terminated	Certification	Masters	Oth
101121	Analyst	CA	Male	New Hire	nanvet	nanracedescription	1	0	0	
101131	Analyst	CA	Male	New Hire	Protected Veteran	nanracedescription	1	0	0	
101142	Analyst	SC	Male	Reorganization	Protected Veteran	nanracedescription	1	0	0	
101146	Coordinator, Human Resources	CA	Female	New Hire	nanvet	nanracedescription	1	0	0	
101164	Analyst	CA	Female	Promotion	Not a protected veteran	Hispanic	1	0	0	

In [27]:
```
finalclean.isna().sum()
```

Out[27]:
```
JobTitle                  0
State                     0
Gender                    0
StartReason               0
VeteranStatusDescription  0
RaceDescription           0
Terminated                0
Certification             0
Masters                   0
Other                     0
Undergraduate             0
TerminationReason         0
pay_cat                   0
dtype: int64
```

In [28]:
```
finalclean
```

Out[28]:

EmployeeNumber	JobTitle	State	Gender	StartReason	VeteranStatusDescription	RaceDescription	Terminated	Certification	Masters
101121	Analyst	CA	Male	New Hire	nanvet	nanracedescription	1	0	0
101131	Analyst	CA	Male	New Hire	Protected Veteran	nanracedescription	1	0	0

101142	Analyst	SC	Male	Reorganization	Protected Veteran	nanracedescription	1	0	0
101146	Coordinator, Human Resources	CA	Female	New Hire	nanvet	nanracedescription	1	0	0
101164	Analyst	CA	Female	Promotion	Not a protected veteran	Hispanic	1	0	0
101167	Analyst	SC	Female	Reorganization	Not a protected veteran	White	0	0	0
101174	Analyst	SC	Male	Reorganization	Protected Veteran	White	1	0	0
101184	Analyst	SC	Male	Reorganization	Not a protected veteran	White	1	0	0
101188	Analyst	CA	nangender	New Hire	Declined to Answer	nanracedescription	1	0	0
101193	Analyst	SC	Female	Reorganization	Not a protected veteran	White	1	0	1
101204	Analyst	SC	Female	New Hire	Not a protected veteran	Black	1	0	0
101205	Analyst	CA	Female	New Hire	Protected Veteran	Hispanic	1	0	0
101207	Analyst	SC	Female	New Hire	Not a protected veteran	White	1	0	0
101217	Analyst	CA	Female	New Hire	Not a protected veteran	Asian	1	0	0
101223	Analyst	CA	Female	New Hire	Not a protected veteran	White	1	0	0
101240	Analyst	VA	Male	New Hire	Protected Veteran	Black	1	1	0
101254	Analyst	CA	Male	New Hire	Protected Veteran	Hispanic	1	0	0
101261	Analyst	CA	Male	New Hire	Not a protected veteran	Prefer Not to Answer	1	0	0
101262	Analyst	CA	Female	New Hire	Protected Veteran	White	1	0	0
101265	Analyst	CA	Female	New Hire	Not a protected veteran	Black	1	0	0
101277	Analyst	CA	Female	New Hire	Not a protected veteran	White	1	0	1
101280	Analyst	CA	Male	New Hire	Protected Veteran	White	0	0	0
101290	Analyst	CA	Male	New Hire	Not a protected veteran	White	0	0	0
101302	Analyst	CA	Female	New Hire	Not a protected veteran	Asian	1	0	0
101311	Accountant	CA	Female	New Hire	Not a protected veteran	Hispanic	0	0	0
101316	Analyst	SC	Female	New Hire	Not a protected veteran	White	0	0	0
101318	Analyst	SC	Male	New Hire	Not a protected veteran	White	0	0	0
101320	Analyst	CA	Female	New Hire	Not a protected veteran	Hispanic	0	0	0
101325	Analyst	SC	Female	New Hire	Not a protected veteran	White	1	1	0
101334	Analyst	CA	Male	New Hire	Not a protected veteran	Native Hawaiian or Other Pacific Islander	0	0	0
101339	Analyst	VA	Female	New Hire	Not a protected veteran	White	1	0	1
101344	Analyst	SC	Female	New Hire	Not a protected veteran	White	0	0	0
101358	Analyst	CA	Female	New Hire	Protected Veteran	Black	1	0	0
101362	Analyst	CA	Male	New Hire	Protected Veteran	Asian	1	0	0
101363	Analyst	SC	Female	New Hire	Not a protected veteran	Hispanic	0	0	0
101375	Analyst	CA	Male	New Hire	Not a protected veteran	Hispanic	1	1	1
101376	Analyst	CA	Male	New Hire	Not a protected veteran	Asian	0	0	0
101380	Analyst	SC	Male	New Hire	Not a protected veteran	White	1	0	0
101382	Analyst	CA	Male	New Hire	Not a protected veteran	White	0	0	0
101395	Analyst	CA	Female	New Hire	Not a protected veteran	Asian	0	0	0
101397	Analyst	CA	Female	New Hire	Not a protected veteran	Asian	0	0	0
101399	Analyst	CA	Male	New Hire	Protected Veteran	White	0	0	0
101404	Analyst	SC	Male	New Hire	Not a protected veteran	White	0	0	0
101414	Analyst	CA	Male	New Hire	Not a protected veteran	Hispanic	0	0	0
101415	Analyst	CA	Male	New Hire	Not a protected veteran	Hispanic	0	0	1
101421	Analyst	CA	Female	New Hire	Not a protected veteran	Asian	0	0	0

EmployeeNumber									
101425	Analyst	VA	Female	New Hire	Not a protected veteran	White	1	0	0
101426	Analyst	VA	Female	New Hire	Not a protected veteran	Black	0	0	0
101427	Administrator, Human Resources	CA	Female	New Hire	Not a protected veteran	Hispanic	0	0	0
101430	Analyst	SC	Male	New Hire	Not a protected veteran	White	0	0	0
101431	Analyst	SC	Female	New Hire	Not a protected veteran	White	0	0	0
101434	Junior Recruiter	CA	Female	New Hire	Not a protected veteran	White	0	1	0
101447	Analyst	VA	Female	New Hire	Not a protected veteran	White	0	0	0
101455	Analyst	CA	Male	New Hire	Not a protected veteran	White	0	0	0
101458	Analyst	VA	Female	New Hire	Protected Veteran	White	0	0	0
101459	Analyst	VA	Male	New Hire	Not a protected veteran	Black	0	0	0
101464	Analyst	CA	Male	New Hire	Not a protected veteran	Hispanic	0	0	0

In [29]:
```
finalclean['TerminationReason'].unique()
```

Out[29]:
```
array(['Resignation', 'Resignation - Career Opportunity',
       'nanterminationreason', 'Resignation - Convert to Government',
       'Layoff', 'Resignation - Personal reasons',
       'Resignation - Relocation', 'Resignation - Management'],
      dtype=object)
```

In [30]:
```
finalclean.columns
```

Out[30]:
```
Index(['JobTitle', 'State', 'Gender', 'StartReason',
       'VeteranStatusDescription', 'RaceDescription', 'Terminated',
       'Certification', 'Masters', 'Other', 'Undergraduate',
       'TerminationReason', 'pay_cat'],
      dtype='object')
```

In [31]:
```
#label encoding is needed for categorical variables to change them to ordinal numerical representations in pay_ca
#OneHotEncoder is needed so the values don't get compared numerically
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
#integer encode
LE = LabelEncoder()
finalclean['Certification'] = LE.fit_transform(finalclean['Certification'])
finalclean['Masters'] = LE.fit_transform(finalclean['Masters'])
finalclean['Other'] = LE.fit_transform(finalclean['Other'])
finalclean['Undergraduate'] = LE.fit_transform(finalclean['Undergraduate'])
finalclean['Terminated'] = LE.fit_transform(finalclean['Terminated'])
finalclean.head()
```

Out[31]:

EmployeeNumber	JobTitle	State	Gender	StartReason	VeteranStatusDescription	RaceDescription	Terminated	Certification	Masters	Oth
101121	Analyst	CA	Male	New Hire	nanvet	nanracedescription	1	0	0	
101131	Analyst	CA	Male	New Hire	Protected Veteran	nanracedescription	1	0	0	
101142	Analyst	SC	Male	Reorganization	Protected Veteran	nanracedescription	1	0	0	
101146	Coordinator, Human Resources	CA	Female	New Hire	nanvet	nanracedescription	1	0	0	
101164	Analyst	CA	Female	Promotion	Not a protected veteran	Hispanic	1	0	0	

In [32]:
```
# convert binary columns to 0 to 1: Terminated

finalclean.loc[(finalclean.Terminated == True), 'Terminated'] = 1
finalclean.loc[(finalclean.Terminated == False), 'Terminated'] = 0
```

In [33]:
```
finalclean.head()
```

Out[33]:

EmployeeNumber	JobTitle	State	Gender	StartReason	VeteranStatusDescription	RaceDescription	Terminated	Certification	Masters	Oth

101121	Analyst	CA	Male	New Hire	nanvet	nanracedescription	1	0	0
101131	Analyst	CA	Male	New Hire	Protected Veteran	nanracedescription	1	0	0
101142	Analyst	SC	Male	Reorganization	Protected Veteran	nanracedescription	1	0	0
101146	Coordinator, Human Resources	CA	Female	New Hire	nanvet	nanracedescription	1	0	0
101164	Analyst	CA	Female	Promotion	Not a protected veteran	Hispanic	1	0	0

In [34]:
```
NoTerm = finalclean.drop(['TerminationReason'], axis = 1)
```

In [35]:
```
# One Hot Encoder for finalclean
ohefinalclean = OneHotEncoder()
finalcleanfeaturearray = ohefinalclean.fit_transform(finalclean[['State', 'Gender', 'StartReason', 'VeteranStatus
```

In [36]:
```
# one hot encoder for NoTerm
ohenoterm = OneHotEncoder()
NoTermfeaturearray = ohenoterm.fit_transform(NoTerm[['State', 'Gender', 'StartReason', 'VeteranStatusDescription'
```

In [37]:
```
finalclean.columns
```

Out[37]:
```
Index(['JobTitle', 'State', 'Gender', 'StartReason',
       'VeteranStatusDescription', 'RaceDescription', 'Terminated',
       'Certification', 'Masters', 'Other', 'Undergraduate',
       'TerminationReason', 'pay_cat'],
      dtype='object')
```

In [38]:
```
NoTerm.columns
```

Out[38]:
```
Index(['JobTitle', 'State', 'Gender', 'StartReason',
       'VeteranStatusDescription', 'RaceDescription', 'Terminated',
       'Certification', 'Masters', 'Other', 'Undergraduate', 'pay_cat'],
      dtype='object')
```

In [39]:
```
ohefinalclean.categories_
```

Out[39]:
```
[array(['CA', 'SC', 'VA'], dtype=object),
 array(['Female', 'Male', 'nangender'], dtype=object),
 array(['New Hire', 'Promotion', 'Reorganization'], dtype=object),
 array(['Declined to Answer', 'Not a protected veteran',
        'Protected Veteran', 'nanvet'], dtype=object),
 array(['Accountant', 'Administrator, Human Resources', 'Analyst',
        'Coordinator, Human Resources', 'Junior Recruiter'], dtype=object),
 array(['Layoff', 'Resignation', 'Resignation - Career Opportunity',
        'Resignation - Convert to Government', 'Resignation - Management',
        'Resignation - Personal reasons', 'Resignation - Relocation',
        'nanterminationreason'], dtype=object),
 array(['average', 'high', 'low'], dtype=object)]
```

In [40]:
```
ohenoterm.categories_
```

Out[40]:
```
[array(['CA', 'SC', 'VA'], dtype=object),
 array(['Female', 'Male', 'nangender'], dtype=object),
 array(['New Hire', 'Promotion', 'Reorganization'], dtype=object),
 array(['Declined to Answer', 'Not a protected veteran',
        'Protected Veteran', 'nanvet'], dtype=object),
 array(['Accountant', 'Administrator, Human Resources', 'Analyst',
        'Coordinator, Human Resources', 'Junior Recruiter'], dtype=object),
 array(['average', 'high', 'low'], dtype=object)]
```

In [41]:
```
finalcleanfeatures = ['CA', 'SC', 'VA','Female', 'Male', 'nangender','New Hire', 'Promotion', 'Reorganization','D
        'Protected Veteran', 'nanvet','Accountant', 'Administrator, Human Resources', 'Analyst',
        'Coordinator, Human Resources', 'Junior Recruiter','Layoff', 'Resignation', 'Resignation - Career Opportu
        'Resignation - Convert to Government', 'Resignation - Management',
        'Resignation - Personal reasons', 'Resignation - Relocation',
        'nanterminationreason','average', 'high', 'low']
```

In [42]:
```
NoTermfeatures = ['CA', 'SC', 'VA','Female', 'Male', 'nangender','New Hire', 'Promotion', 'Reorganization','Decli
        'Protected Veteran', 'nanvet','Accountant', 'Administrator, Human Resources', 'Analyst',
        'Coordinator, Human Resources', 'Junior Recruiter', 'average', 'high', 'low']
```

In [43]:
```
featuresfinalclean = pd.DataFrame(finalcleanfeaturearray, columns = finalcleanfeatures)
```

In [44]:
```
featuresNoTerm = pd.DataFrame(NoTermfeaturearray, columns = NoTermfeatures)
```

In [45]:
```
num = finalclean[['Certification', 'Masters', 'Other', 'Undergraduate', 'Terminated']]
```

In [46]:
```
featfinalclean = featuresfinalclean.reset_index()
featNoTerm = featuresNoTerm.reset_index()
num = num.reset_index()
```

In [47]:
```
cleanjrquit = pd.concat([featfinalclean, num], axis = 1)
cleanjrquit.head()
```

Out[47]:

	index	CA	SC	VA	Female	Male	nangender	New Hire	Promotion	Reorganization	...	nanterminationreason	average	high	low	EmployeeNum
0	0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	10
1	1	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	10
2	2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	1.0	0.0	0.0	1.0	10
3	3	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	1.0	10
4	4	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	0.0	1.0	10

5 rows × 36 columns

In [48]:
```
cleanNoTerm = pd.concat([featNoTerm, num], axis = 1)
cleanNoTerm.head()
```

Out[48]:

	index	CA	SC	VA	Female	Male	nangender	New Hire	Promotion	Reorganization	...	Junior Recruiter	average	high	low	EmployeeNumber	Certifi
0	0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	101121	
1	1	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	101131	
2	2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	1.0	101142	
3	3	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	1.0	101146	
4	4	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	101164	

5 rows × 28 columns

In [49]:
```
cleanjrquit.columns
```

Out[49]:
```
Index(['index', 'CA', 'SC', 'VA', 'Female', 'Male', 'nangender', 'New Hire',
       'Promotion', 'Reorganization', 'Declined to Answer',
       'Not a protected veteran', 'Protected Veteran', 'nanvet', 'Accountant',
       'Administrator, Human Resources', 'Analyst',
       'Coordinator, Human Resources', 'Junior Recruiter', 'Layoff',
       'Resignation', 'Resignation - Career Opportunity',
       'Resignation - Convert to Government', 'Resignation - Management',
       'Resignation - Personal reasons', 'Resignation - Relocation',
       'nanterminationreason', 'average', 'high', 'low', 'EmployeeNumber',
       'Certification', 'Masters', 'Other', 'Undergraduate', 'Terminated'],
      dtype='object')
```

In [50]:
```
#dropping columns with no meaning, index, nangender, declined to answer, did not return form, nanvet, nanterminat

cleanjrquit = cleanjrquit.drop(['index', 'nangender', 'nanterminationreason','nanvet', 'Declined to Answer', 'Emp
```

In [51]:
```
cleanNoTerm.columns
```

Out[51]:
```
Index(['index', 'CA', 'SC', 'VA', 'Female', 'Male', 'nangender', 'New Hire',
```

Out[51]:
```
       'Promotion', 'Reorganization', 'Declined to Answer',
       'Not a protected veteran', 'Protected Veteran', 'nanvet', 'Accountant',
       'Administrator, Human Resources', 'Analyst',
       'Coordinator, Human Resources', 'Junior Recruiter', 'average', 'high',
       'low', 'EmployeeNumber', 'Certification', 'Masters', 'Other',
       'Undergraduate', 'Terminated'],
      dtype='object')
```

In [52]:
```
#dropping columns with no meaning, index, nangender, declined to answer, did not return form, nanvet, employee nu

cleanNoTerm = cleanNoTerm.drop(['index', 'nangender', 'nanvet', 'Declined to Answer', 'EmployeeNumber'], axis = 1
```

In [53]:
```
y = cleanjrquit.Terminated
```

In [54]:
```
predictorsjrquit = list(cleanjrquit.columns)
outcome = 'Terminated'
predictorsjrquit.remove(outcome)
Xjrquit = cleanjrquit[predictorsjrquit]
```

In [55]:
```
Xjrquit.head()
```

Out[55]:

	CA	SC	VA	Female	Male	New Hire	Promotion	Reorganization	Not a protected veteran	Protected Veteran	...	Resignation - Management	Resignation - Personal reasons	Resignation - Relocation	average	high
0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0
1	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0

5 rows × 29 columns

In [56]:
```
predictorsnoterm = list(cleanNoTerm.columns)
outcome = 'Terminated'
predictorsnoterm.remove(outcome)
Xnoterm = cleanNoTerm[predictorsnoterm]
```

In [57]:
```
#split for cleanjrquit
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xjrquit,y, test_size = 0.2, random_state = 30)
```

In [58]:
```
#split for cleanNoTerm
from sklearn.model_selection import train_test_split
X_trainNoTerm, X_testNoTerm, y_trainNoTerm, y_testNoTerm = train_test_split(Xnoterm,y, test_size = 0.2, random_st
```

In [59]:
```
X_train.shape
```

Out[59]:
```
(45, 29)
```

In [60]:
```
X_test.shape
```

Out[60]:
```
(12, 29)
```

In [61]:
```
X_trainNoTerm.shape
```

Out[61]:
```
(45, 22)
```

In [62]:
```
X_testNoTerm.shape
```

Out[62]:
```
(12, 22)
```

Classification

In [63]:
```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, plot_confusion_matrix, classification_report, accuracy_score
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from dmba import classificationSummary, gainsChart
```

In [64]:
```
#Function for printing model evaluation metrics

classes = ('Satisfied', 'Unsatisfied')

def stat_print(y_train, pred_cancel):
    print('Recall Score : ',recall_score(y_train, pred_cancel, average='weighted'))
    print('Accuracy Score : ',accuracy_score(y_train, pred_cancel))
    print('F1 Score : ',f1_score(y_train,pred_cancel))
    print('Precision Score : ',precision_score(y_train,pred_cancel))


def confusionMatrices(model, title):
    print(title + ' - training results')
    classificationSummary(y_train, model.predict(X_train), class_names=classes)
    print(title + ' - validation results')
    valid_pred = model.predict(X_test)
    classificationSummary(y_test, valid_pred, class_names=classes)
```

Data Science Jr Quit (With Termination)

In [65]:
```
# logistic regression jrquit
logit = LogisticRegressionCV(penalty='l2', solver='saga', cv=5, max_iter=110000, random_state=3).fit(X_train, y_t

logit_confusion = confusionMatrices(logit, 'Logistic Regression')
```

```
Logistic Regression - training results
Confusion Matrix (Accuracy 1.0000)

            Prediction
    Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           0          23
Logistic Regression - validation results
Confusion Matrix (Accuracy 0.6667)

            Prediction
    Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           4           1
```

In [66]:
```
#decision tree
#Unprunned decision tree
dtree = DecisionTreeClassifier(random_state=7).fit(X_train, y_train)
tree_confusion = confusionMatrices(dtree, 'Decision Tree')
```

```
Decision Tree - training results
Confusion Matrix (Accuracy 1.0000)

            Prediction
    Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           0          23
Decision Tree - validation results
Confusion Matrix (Accuracy 0.7500)

            Prediction
    Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           3           2
```

In [67]:
```
#bagging
#Using the decision classification tree as the base estimator, bagging is used to improve metrics
bagging = BaggingClassifier(dtree, random_state=3, max_samples = 0.5, max_features = 0.5)
bagging.fit(X_train, y_train)
bag_confusion = confusionMatrices(bagging, 'Bagging')
```

```
Bagging - training results
Confusion Matrix (Accuracy 0.8444)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          21           1
Unsatisfied           6          17
Bagging - validation results
Confusion Matrix (Accuracy 0.7500)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           3           2
```

In [68]:
```
#adaboost
#Using the decision classification tree as the base estimator, adaboost is used to improve metrics
adaboost = AdaBoostClassifier(n_estimators = 100, base_estimator = dtree, random_state=3)
adaboost.fit(X_train, y_train)
ada_confusion = confusionMatrices(adaboost, 'Adaboost')
```

```
Adaboost - training results
Confusion Matrix (Accuracy 1.0000)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           0          23
Adaboost - validation results
Confusion Matrix (Accuracy 0.7500)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           3           2
```

In [69]:
```
#random forest
rf = RandomForestClassifier(n_estimators=100, random_state=3).fit(X_train, y_train.values.ravel())
forest_confusion = confusionMatrices(rf, 'Random Forest')
```

```
Random Forest - training results
Confusion Matrix (Accuracy 1.0000)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           0          23
Random Forest - validation results
Confusion Matrix (Accuracy 0.7500)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           3           2
```

In [70]:
```
#nn
#scaling training features to 1.0
scaleInput = MinMaxScaler()
scaleInput.fit(X_train * 1.0)
#hidden_layer_sizes = 10: number of neurons in the ith hidden layer
#activation: logistic sigmoid function
#solver: weight optimization using stocastic gradient descent
neuralNet = MLPClassifier(hidden_layer_sizes = (10),
 activation = 'logistic',
 solver = 'sgd',
 max_iter = 3000,
 random_state = 1)
neuralNet.fit(scaleInput.transform(X_train), y_train)
NNConfusion = confusionMatrices(neuralNet, 'Neural Network')
```

```
Neural Network - training results
Confusion Matrix (Accuracy 0.5111)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           0          22
Unsatisfied           0          23
Neural Network - validation results
Confusion Matrix (Accuracy 0.4167)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           0           7
Unsatisfied           0           5
```

Results

In [71]:
```
#Decision Tree
dtree_pred_test = dtree.predict(X_test)
print('Accuracy score:')
print(accuracy_score(y_test,dtree_pred_test))
dtree_roc = metrics.roc_curve(y_test, dtree_pred_test)
dtree_auc = metrics.auc(dtree_roc[0], dtree_roc[1])
dtree_plot = metrics.RocCurveDisplay(dtree_roc[0], dtree_roc[1],
roc_auc=dtree_auc, estimator_name='Decision Tree')
```

```
Accuracy score:
0.75
```

In [72]:
```
print(classification_report(y_test, dtree_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82         7
           1       1.00      0.40      0.57         5

    accuracy                           0.75        12
   macro avg       0.85      0.70      0.70        12
weighted avg       0.82      0.75      0.72        12
```

In [73]:
```
#Bagging
bagging_pred_test = bagging.predict(X_test)
print('Accuracy score:')
print(accuracy_score(y_test,bagging_pred_test))
bagging_roc = metrics.roc_curve(y_test, bagging_pred_test)
bagging_auc = metrics.auc(bagging_roc[0], bagging_roc[1])
bagging_plot = metrics.RocCurveDisplay(bagging_roc[0], bagging_roc[1],
roc_auc=bagging_auc, estimator_name='Bagging')
```

```
Accuracy score:
0.75
```

In [74]:
```
print(classification_report(y_test, bagging_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82         7
           1       1.00      0.40      0.57         5

    accuracy                           0.75        12
   macro avg       0.85      0.70      0.70        12
weighted avg       0.82      0.75      0.72        12
```

In [75]:
```
#Adaboost
adaboost_pred_test = adaboost.predict(X_test)
print(accuracy_score(y_test,adaboost_pred_test))
adaboost_roc = metrics.roc_curve(y_test, adaboost_pred_test)
adaboost_auc = metrics.auc(adaboost_roc[0], adaboost_roc[1])
adaboost_plot = metrics.RocCurveDisplay(adaboost_roc[0], adaboost_roc[1],
```

```
roc_auc=adaboost_auc, estimator_name='Adaboost')
```

```
0.75
```

In [76]:
```
print(classification_report(y_test, adaboost_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82         7
           1       1.00      0.40      0.57         5

    accuracy                           0.75        12
   macro avg       0.85      0.70      0.70        12
weighted avg       0.82      0.75      0.72        12
```

In [77]:
```
#Logistic Regression
logit_pred_test = logit.predict(X_test)
print(accuracy_score(y_test,logit_pred_test))
logit_roc = metrics.roc_curve(y_test, logit_pred_test)
logit_auc = metrics.auc(logit_roc[0], logit_roc[1])
logit_plot = metrics.RocCurveDisplay(logit_roc[0], logit_roc[1],
roc_auc=logit_auc, estimator_name='Logistic Regression')
```

```
0.6666666666666666
```

In [78]:
```
print(classification_report(y_test, logit_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.64      1.00      0.78         7
           1       1.00      0.20      0.33         5

    accuracy                           0.67        12
   macro avg       0.82      0.60      0.56        12
weighted avg       0.79      0.67      0.59        12
```

In [79]:
```
#Random Forest
rf_pred_test = rf.predict(X_test)
print(accuracy_score(y_test,rf_pred_test))
rf_roc = metrics.roc_curve(y_test, rf_pred_test)
rf_auc = metrics.auc(rf_roc[0], rf_roc[1])
rf_plot = metrics.RocCurveDisplay(rf_roc[0], logit_roc[1],
roc_auc=rf_auc, estimator_name='Random Forest')
```

```
0.75
```

In [80]:
```
print(classification_report(y_test, rf_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82         7
           1       1.00      0.40      0.57         5

    accuracy                           0.75        12
   macro avg       0.85      0.70      0.70        12
weighted avg       0.82      0.75      0.72        12
```

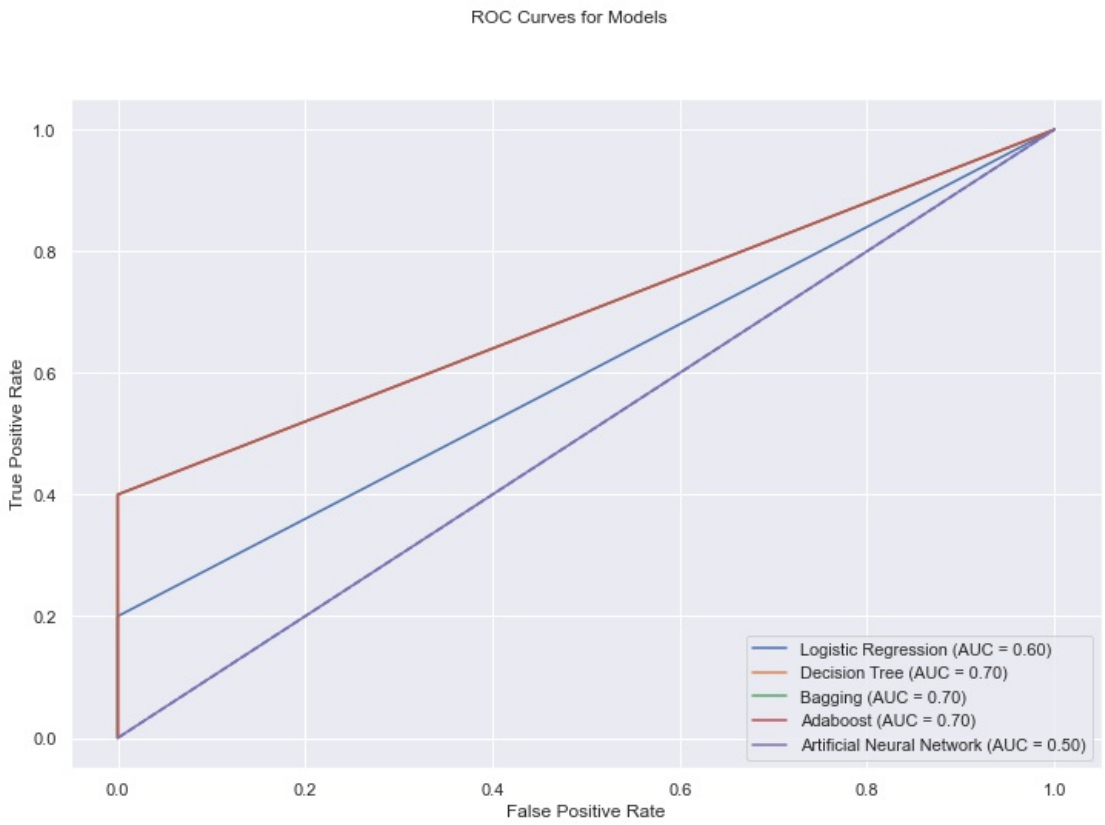
In [81]:
```
#Neural Network
neuralNet_pred_test = neuralNet.predict(X_test)
print(accuracy_score(y_test,neuralNet_pred_test))
neuralNet_roc = metrics.roc_curve(y_test, neuralNet_pred_test)
neuralNet_auc = metrics.auc(neuralNet_roc[0], neuralNet_roc[1])
neuralNet_plot = metrics.RocCurveDisplay(neuralNet_roc[0], neuralNet_roc[1],
roc_auc=neuralNet_auc, estimator_name='Artificial Neural Network')
```

```
0.4166666666666667
```

In [82]:
```
print(classification_report(y_test, neuralNet_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.42      1.00      0.59         5

    accuracy                           0.42        12
   macro avg       0.21      0.50      0.29        12
weighted avg       0.17      0.42      0.25        12
```

In [83]:
```
# Plotting ROC Curves for models
fig, ax = plt.subplots(figsize=(12,8))
fig.suptitle('ROC Curves for Models', fontsize=12)
plt.plot([0, 1], [0, 1], linestyle = '--', color = '#174ab0')
plt.xlabel('',fontsize=12)
plt.ylabel('',fontsize=12)
logit_plot.plot(ax)
dtree_plot.plot(ax)
bagging_plot.plot(ax)
adaboost_plot.plot(ax)
neuralNet_plot.plot(ax)
plt.show()
```



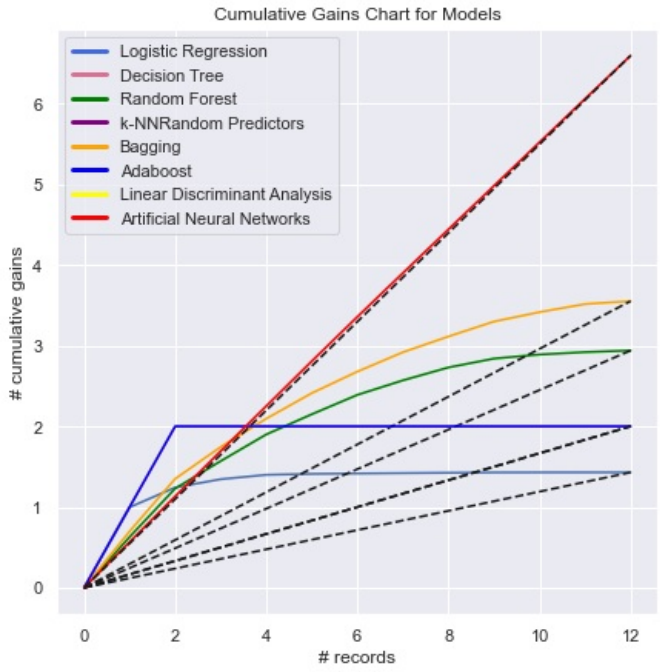
In [84]:
```
rf_plot.plot(ax)
```

Out[84]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2255756aaf0>
```

In [85]:
```
from dmba import liftChart
from matplotlib.lines import Line2D
logisticchart = pd.Series(logit.predict_proba(X_test)[:, 1])
logisticchart = logisticchart.sort_values(ascending=False)
dtreechart = pd.Series(dtree.predict_proba(X_test)[:, 1])
dtreechart = dtreechart.sort_values(ascending=False)
rfchart = pd.Series(rf.predict_proba(X_test)[:, 1])
rfchart = rfchart.sort_values(ascending=False)
baggingchart = pd.Series(bagging.predict_proba(X_test)[:, 1])
baggingchart = baggingchart.sort_values(ascending=False)
adaboostchart = pd.Series(adaboost.predict_proba(X_test)[:, 1])
```

```
adaboostchart = adaboostchart.sort_values(ascending=False)
neuralNetchart = pd.Series(neuralNet.predict_proba((X_test))[:, 1])
neuralNetchart = neuralNetchart.sort_values(ascending=False)
ax= gainsChart(logisticchart, figsize=[7,7])
gainsChart(dtreechart, color='palevioletred', ax=ax)
gainsChart(rfchart, color='green', ax=ax)
gainsChart(baggingchart, color='orange', ax=ax)
gainsChart(adaboostchart, color='blue', ax=ax)
gainsChart(neuralNetchart, color='red', ax=ax)
ax.set_title('Cumulative Gains Chart for Models')
colors = ['royalblue', 'palevioletred', 'green', 'purple', 'orange', 'blue', 'yellow', 'red']
lines = [Line2D([0], [0], color=c, linewidth=3, linestyle='-') for c in colors]
labels=['Logistic Regression', 'Decision Tree', 'Random Forest', 'k-NN'
 'Random Predictors', 'Bagging', 'Adaboost', 'Linear Discriminant Analysis', 'Artificial Neural Networks']
plt.legend(lines, labels)
```

Out[85]:
```
<matplotlib.legend.Legend at 0x225576b20a0>
```



No Termination

In [86]:
```
#Function for printing model evaluation metrics

classes = ('Satisfied', 'Unsatisfied')

def stat_print(y_train, pred_cancel):
    print('Recall Score : ',recall_score(y_trainNoTerm, pred_cancel, average='weighted'))
    print('Accuracy Score : ',accuracy_score(y_trainNoTerm, pred_cancel))
    print('F1 Score : ',f1_score(y_trainNoTerm,pred_cancel))
    print('Precision Score : ',precision_score(y_trainNoTerm,pred_cancel))


def confusionMatrices(model, title):
    print(title + ' - training results')
    classificationSummary(y_trainNoTerm, model.predict(X_trainNoTerm), class_names=classes)
    print(title + ' - validation results')
    valid_pred = model.predict(X_testNoTerm)
    classificationSummary(y_testNoTerm, valid_pred, class_names=classes)
```

In [87]:
```
# logistic regression jrquit
logitNT = LogisticRegressionCV(penalty='l2', solver='saga', cv=5, max_iter=110000, random_state=3).fit(X_trainNo

logit_confusionNT = confusionMatrices(logitNT, 'Logistic Regression')
```

```
Logistic Regression - training results
Confusion Matrix (Accuracy 0.7778)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          17           5
Unsatisfied           5          18
Logistic Regression - validation results
Confusion Matrix (Accuracy 0.6667)

            Prediction
```

```
     Actual   Satisfied Unsatisfied
  Satisfied           6           1
Unsatisfied           3           2
```

In [88]:

```
#decision tree
#Unprunned decision tree
dtreeNT = DecisionTreeClassifier(random_state=7).fit(X_trainNoTerm, y_trainNoTerm)
tree_confusionNT = confusionMatrices(dtreeNT, 'Decision Tree')
```

```
Decision Tree - training results
Confusion Matrix (Accuracy 0.9778)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           1          22
Decision Tree - validation results
Confusion Matrix (Accuracy 0.5833)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           5           2
Unsatisfied           3           2
```

In [89]:

```
#bagging
#Using the decision classification tree as the base estimator, bagging is used to improve metrics
baggingNT = BaggingClassifier(dtreeNT, random_state=3, max_samples = 0.5, max_features = 0.5)
baggingNT.fit(X_trainNoTerm, y_trainNoTerm)
bag_confusionNT = confusionMatrices(baggingNT, 'Bagging')
```

```
Bagging - training results
Confusion Matrix (Accuracy 0.8222)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          19           3
Unsatisfied           5          18
Bagging - validation results
Confusion Matrix (Accuracy 0.6667)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           6           1
Unsatisfied           3           2
```

In [90]:

```
#adaboost
#Using the decision classification tree as the base estimator, adaboost is used to improve metrics
adaboostNT = AdaBoostClassifier(n_estimators = 100, base_estimator = dtreeNT, random_state=3)
adaboostNT.fit(X_trainNoTerm, y_trainNoTerm)
ada_confusionNT = confusionMatrices(adaboostNT, 'Adaboost')
```

```
Adaboost - training results
Confusion Matrix (Accuracy 0.9778)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied           1          22
Adaboost - validation results
Confusion Matrix (Accuracy 0.5833)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           5           2
Unsatisfied           3           2
```

In [91]:

```
#random forest
rfNT = RandomForestClassifier(n_estimators=100, random_state=3).fit(X_trainNoTerm, y_trainNoTerm.values.ravel())
forest_confusionNT = confusionMatrices(rfNT, 'Random Forest')
```

```
Random Forest - training results
Confusion Matrix (Accuracy 0.9778)

            Prediction
     Actual   Satisfied Unsatisfied
```

```
  Satisfied          22           0
Unsatisfied           1          22
Random Forest - validation results
Confusion Matrix (Accuracy 0.6667)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           6           1
Unsatisfied           3           2
```

In [92]:

```
#nn
#scaling training features to 1.0
scaleInputNT = MinMaxScaler()
scaleInput.fit(X_trainNoTerm * 1.0)
#hidden_layer_sizes = 10: number of neurons in the ith hidden layer
#activation: logistic sigmoid function
#solver: weight optimization using stocastic gradient descent
neuralNetNT = MLPClassifier(hidden_layer_sizes = (10),
 activation = 'logistic',
 solver = 'sgd',
 max_iter = 3000,
 random_state = 1)
neuralNetNT.fit(scaleInput.transform(X_trainNoTerm), y_trainNoTerm)
NNConfusionNT = confusionMatrices(neuralNetNT, 'Neural Network')
```

```
Neural Network - training results
Confusion Matrix (Accuracy 0.4889)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied          22           0
Unsatisfied          23           0
Neural Network - validation results
Confusion Matrix (Accuracy 0.5833)

            Prediction
     Actual   Satisfied Unsatisfied
  Satisfied           7           0
Unsatisfied           5           0
```

Results

In [93]:

```
#Decision Tree
dtree_pred_testNT = dtreeNT.predict(X_testNoTerm)
print('Accuracy score:')
print(accuracy_score(y_testNoTerm,dtree_pred_testNT))
dtree_rocNT = metrics.roc_curve(y_testNoTerm, dtree_pred_testNT)
dtree_aucNT = metrics.auc(dtree_rocNT[0], dtree_rocNT[1])
dtree_plotNT = metrics.RocCurveDisplay(dtree_rocNT[0], dtree_rocNT[1],
roc_auc=dtree_aucNT, estimator_name='Decision Tree')
```

```
Accuracy score:
0.5833333333333334
```

In [94]:

```
print(classification_report(y_testNoTerm, dtree_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.62      0.71      0.67         7
           1       0.50      0.40      0.44         5

    accuracy                           0.58        12
   macro avg       0.56      0.56      0.56        12
weighted avg       0.57      0.58      0.57        12
```

In [95]:

```
#Bagging
bagging_pred_testNT = baggingNT.predict(X_testNoTerm)
print('Accuracy score:')
print(accuracy_score(y_testNoTerm,bagging_pred_testNT))
bagging_rocNT = metrics.roc_curve(y_testNoTerm, bagging_pred_testNT)
bagging_aucNT = metrics.auc(bagging_rocNT[0], bagging_rocNT[1])
bagging_plotNT = metrics.RocCurveDisplay(bagging_rocNT[0], bagging_rocNT[1],
roc_auc=bagging_auc, estimator_name='Bagging')
```

```
Accuracy score:
0.6666666666666666
```

In [96]:

```
print(classification_report(y_testNoTerm, bagging_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.67      0.86      0.75         7
           1       0.67      0.40      0.50         5

    accuracy                           0.67        12
   macro avg       0.67      0.63      0.62        12
weighted avg       0.67      0.67      0.65        12
```

In [97]:

```
#Adaboost
adaboost_pred_testNT = adaboostNT.predict(X_testNoTerm)
print(accuracy_score(y_testNoTerm,adaboost_pred_testNT))
adaboost_rocNT = metrics.roc_curve(y_testNoTerm, adaboost_pred_testNT)
adaboost_aucNT = metrics.auc(adaboost_rocNT[0], adaboost_rocNT[1])
adaboost_plotNT = metrics.RocCurveDisplay(adaboost_rocNT[0], adaboost_rocNT[1],
roc_auc=adaboost_aucNT, estimator_name='Adaboost')
```

```
0.5833333333333334
```

In [98]:

```
print(classification_report(y_testNoTerm, adaboost_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.62      0.71      0.67         7
           1       0.50      0.40      0.44         5

    accuracy                           0.58        12
   macro avg       0.56      0.56      0.56        12
weighted avg       0.57      0.58      0.57        12
```

In [99]:

```
#Logistic Regression
logit_pred_testNT = logitNT.predict(X_testNoTerm)
print(accuracy_score(y_testNoTerm,logit_pred_testNT))
logit_rocNT = metrics.roc_curve(y_testNoTerm, logit_pred_testNT)
logit_aucNT = metrics.auc(logit_rocNT[0], logit_rocNT[1])
logit_plotNT = metrics.RocCurveDisplay(logit_rocNT[0], logit_rocNT[1],
roc_auc=logit_aucNT, estimator_name='Logistic Regression')
```

```
0.6666666666666666
```

In [100…

```
print(classification_report(y_testNoTerm, logit_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.67      0.86      0.75         7
           1       0.67      0.40      0.50         5

    accuracy                           0.67        12
   macro avg       0.67      0.63      0.62        12
weighted avg       0.67      0.67      0.65        12
```

In [101…

```
#Random Forest
rf_pred_testNT = rfNT.predict(X_testNoTerm)
print(accuracy_score(y_testNoTerm,rf_pred_testNT))
rf_rocNT = metrics.roc_curve(y_testNoTerm, rf_pred_testNT)
rf_aucNT = metrics.auc(rf_rocNT[0], rf_rocNT[1])
rf_plotNT = metrics.RocCurveDisplay(rf_rocNT[0], logit_rocNT[1],
roc_auc=rf_aucNT, estimator_name='Random Forest')
```

```
0.6666666666666666
```

In [102…

```
print(classification_report(y_testNoTerm, rf_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.67      0.86      0.75         7
           1       0.67      0.40      0.50         5

    accuracy                           0.67        12
   macro avg       0.67      0.63      0.62        12
weighted avg       0.67      0.67      0.65        12
```

In [103…

```
#Neural Network
neuralNet_pred_testNT = neuralNetNT.predict(X_testNoTerm)
print(accuracy_score(y_testNoTerm,neuralNet_pred_testNT))
neuralNet_rocNT = metrics.roc_curve(y_testNoTerm, neuralNet_pred_testNT)
neuralNet_aucNT = metrics.auc(neuralNet_rocNT[0], neuralNet_rocNT[1])
neuralNet_plotNT = metrics.RocCurveDisplay(neuralNet_rocNT[0], neuralNet_rocNT[1],
roc_auc=neuralNet_aucNT, estimator_name='Artificial Neural Network')
```

```
0.5833333333333334
```

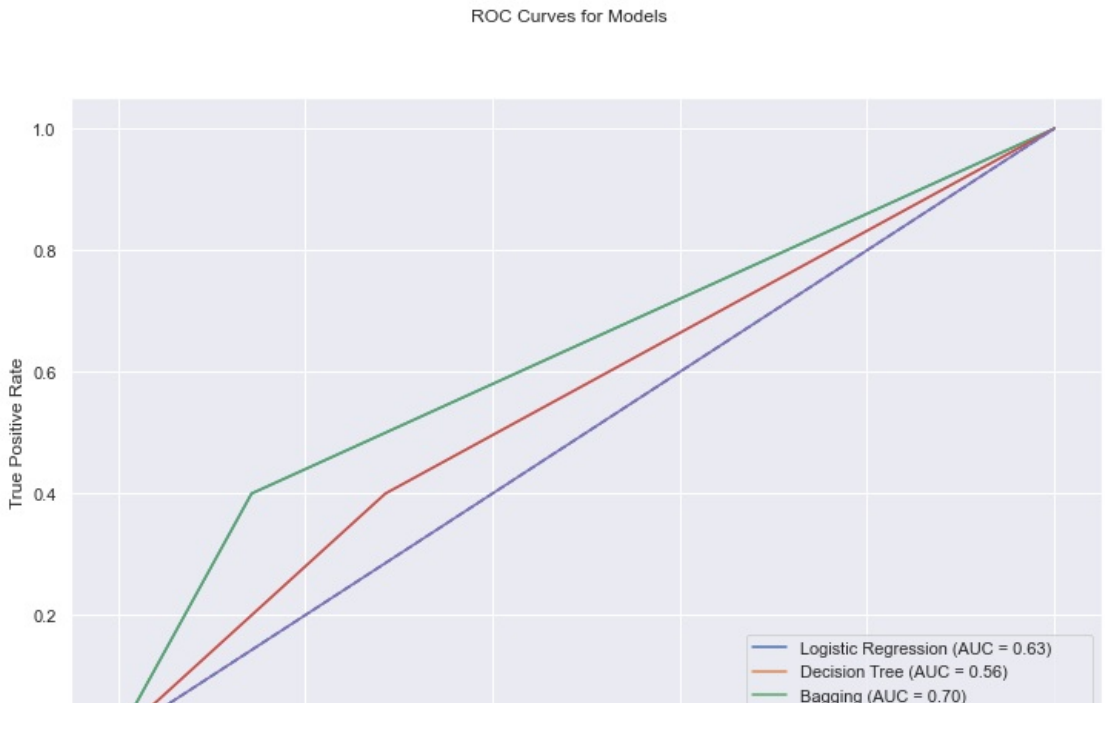
In [104…

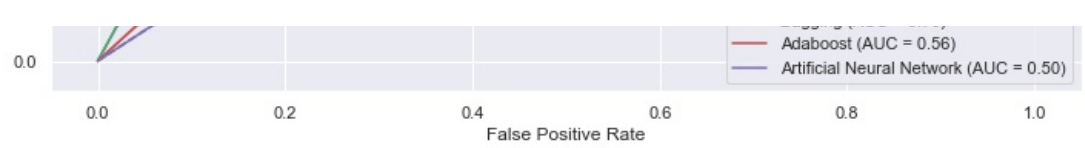
```
print(classification_report(y_testNoTerm, neuralNet_pred_testNT))
```

```
              precision    recall  f1-score   support

           0       0.58      1.00      0.74         7
           1       0.00      0.00      0.00         5

    accuracy                           0.58        12
   macro avg       0.29      0.50      0.37        12
weighted avg       0.34      0.58      0.43        12
```

In [105…

```
# Plotting ROC Curves for models
fig, ax = plt.subplots(figsize=(12,8))
fig.suptitle('ROC Curves for Models', fontsize=12)
plt.plot([0, 1], [0, 1], linestyle = '--', color = '#174ab0')
plt.xlabel('',fontsize=12)
plt.ylabel('',fontsize=12)
logit_plotNT.plot(ax)
dtree_plotNT.plot(ax)
bagging_plotNT.plot(ax)
adaboost_plotNT.plot(ax)
neuralNet_plotNT.plot(ax)
plt.show()
```



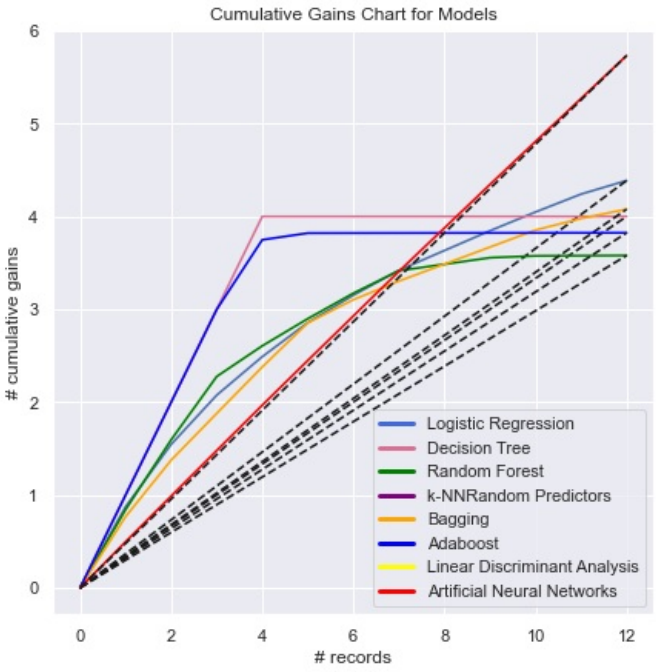


In [109…

```
from dmba import liftChart
from matplotlib.lines import Line2D
logisticchartNT = pd.Series(logitNT.predict_proba(X_testNoTerm)[:, 1])
logisticchartNT = logisticchartNT.sort_values(ascending=False)
dtreechartNT = pd.Series(dtreeNT.predict_proba(X_testNoTerm)[:, 1])
dtreechartNT = dtreechartNT.sort_values(ascending=False)
rfchartNT = pd.Series(rfNT.predict_proba(X_testNoTerm)[:, 1])
rfchartNT = rfchartNT.sort_values(ascending=False)
baggingchartNT = pd.Series(baggingNT.predict_proba(X_testNoTerm)[:, 1])
baggingchartNT = baggingchartNT.sort_values(ascending=False)
adaboostchartNT = pd.Series(adaboostNT.predict_proba(X_testNoTerm)[:, 1])
adaboostchartNT = adaboostchartNT.sort_values(ascending=False)
neuralNetchartNT = pd.Series(neuralNetNT.predict_proba(scaleInput.transform(X_testNoTerm))[:, 1])
neuralNetchartNT = neuralNetchartNT.sort_values(ascending=False)
ax= gainsChart(logisticchartNT, figsize=[7,7])
gainsChart(dtreechartNT, color='palevioletred', ax=ax)
gainsChart(rfchartNT, color='green', ax=ax)
gainsChart(baggingchartNT, color='orange', ax=ax)
gainsChart(adaboostchartNT, color='blue', ax=ax)
gainsChart(neuralNetchartNT, color='red', ax=ax)
ax.set_title('Cumulative Gains Chart for Models')
colors = ['royalblue', 'palevioletred', 'green', 'purple', 'orange', 'blue', 'yellow', 'red']
lines = [Line2D([0], [0], color=c, linewidth=3, linestyle='-') for c in colors]
labels=['Logistic Regression', 'Decision Tree', 'Random Forest', 'k-NN'
 'Random Predictors', 'Bagging', 'Adaboost', 'Linear Discriminant Analysis', 'Artificial Neural Networks']
plt.legend(lines, labels)
```

Out[109…

```
<matplotlib.legend.Legend at 0x225576758b0>
```



Feature Importance without Termination Reason

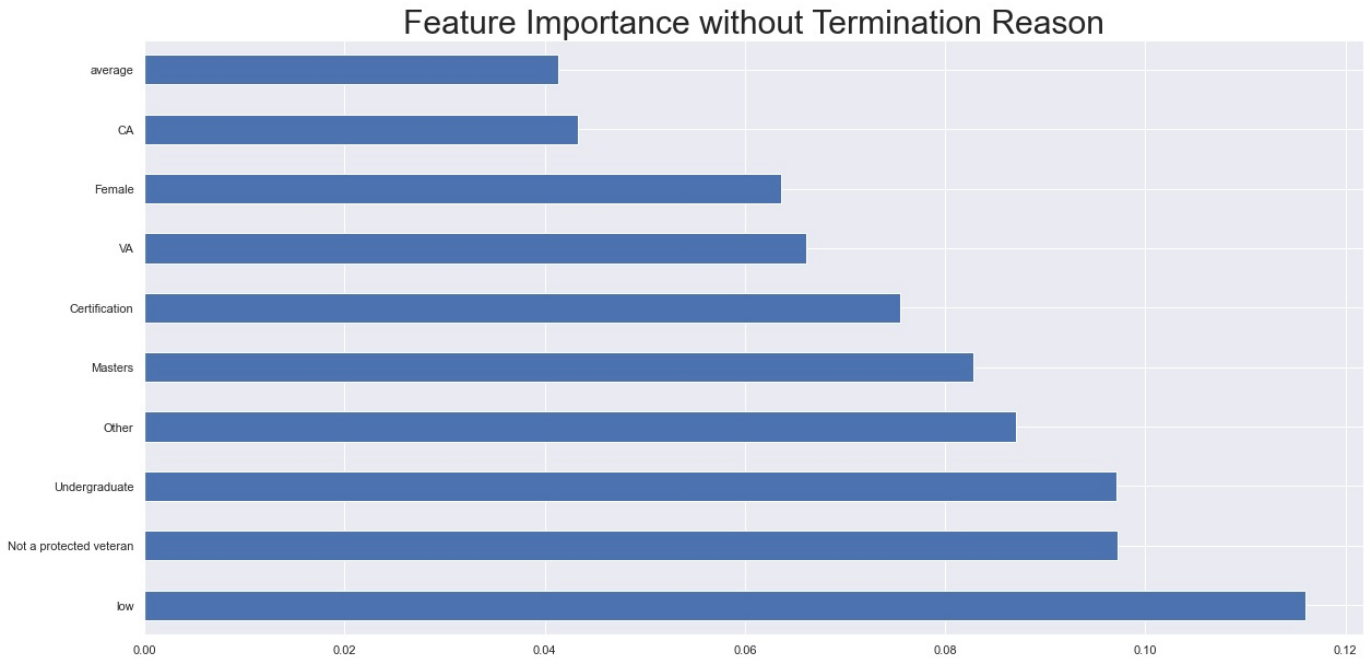
In [110…

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

clf = BaggingClassifier(DecisionTreeClassifier())
clf.fit(Xnoterm, y)

feature_importances = np.mean([
    tree.feature_importances_ for tree in clf.estimators_
], axis=0)
```

In [111…

```
feat_imp = pd.Series(feature_importances, index = X_trainNoTerm.columns)
feat_imp.nlargest(10).plot(kind = 'barh')
plt.rc('xtick', labelsize = 15)
plt.rc('ytick', labelsize = 15)
```

```
plt.rc('axes', titlesize = 30)
plt.title('Feature Importance without Termination Reason')
plt.show()
```



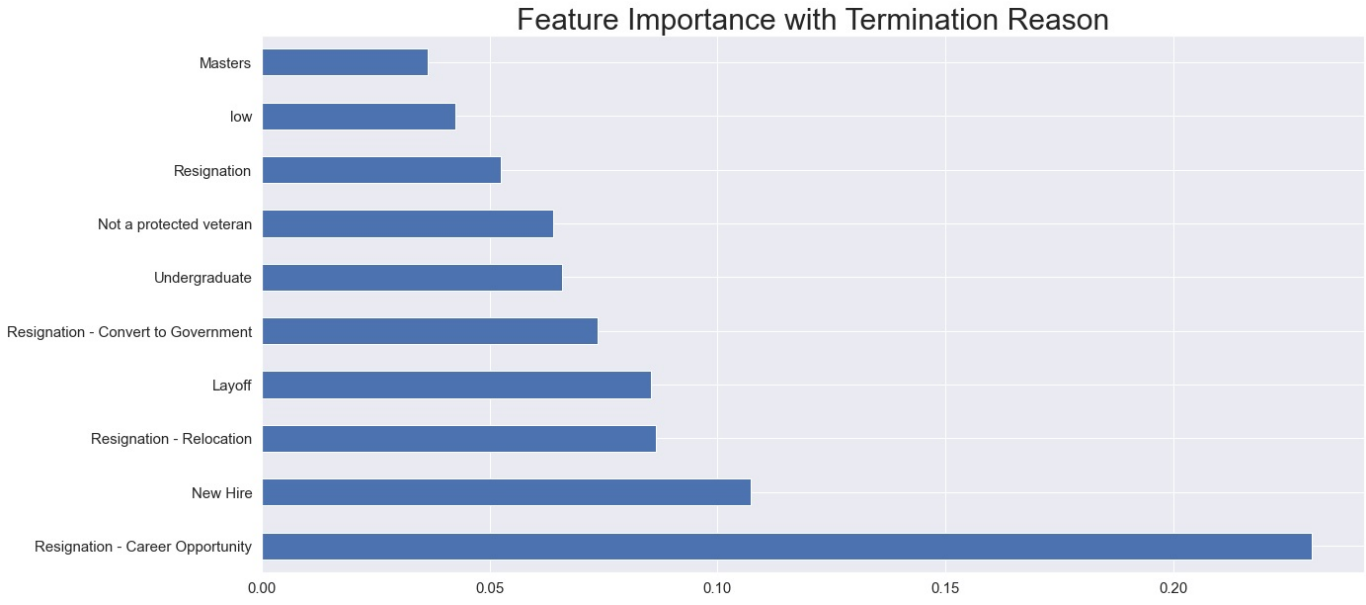
Feature Importance with Termination Reason

In [121…

```
from sklearn.ensemble import AdaBoostClassifier

clf = BaggingClassifier(DecisionTreeClassifier())
clf.fit(Xjrquit, y)

feature_importances = np.mean([
    tree.feature_importances_ for tree in clf.estimators_
], axis=0)
```

In [122…

```
feat_imp = pd.Series(feature_importances, index = X_train.columns)
feat_imp.nlargest(10).plot(kind = 'barh')
plt.rc('xtick', labelsize = 15)
plt.rc('ytick', labelsize = 15)
plt.rc('axes', titlesize = 30)
plt.title('Feature Importance with Termination Reason')
plt.show()
```



In [ ]: