Data Challenge 1

February 20, 2019

1 Jeremy Ferlic - Data Challenge #1

1.1 Imports

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

1.2 Load in Data

```
In [2]: # Read in data frame
        dat = pd.read_csv("employee_retention_data.csv")

        # Massage date-time objects and categorize department into dept_id
        dat['join_date'] = pd.to_datetime(dat['join_date'])
        dat['quit_date'] = pd.to_datetime(dat['quit_date'])
        dat['dept'] = dat['dept'].astype('category')
        dat['dept_id'] = dat['dept'].cat.codes

        # Print out some basic information about the dataset
        print(dat.shape)
        print()
        print(dat.dtypes)
        print()
        print(dat.head())
```

```
(24702, 8)

employee_id           float64
company_id              int64
dept                 category
seniority               int64
salary                float64
join_date      datetime64[ns]
quit_date      datetime64[ns]
```

```
dept_id                   int8
dtype: object

   employee_id  company_id              dept  seniority    salary  join_date  \
0      13021.0           7  customer_service         28   89000.0 2014-03-24
1     825355.0           7         marketing         20  183000.0 2013-04-29
2     927315.0           4         marketing         14  101000.0 2014-10-13
3     662910.0           7  customer_service         20  115000.0 2012-05-14
4     256971.0           2      data_science         23  276000.0 2011-10-17

   quit_date  dept_id
0 2015-10-30        0
1 2014-04-04        4
2        NaT        4
3 2013-06-07        0
4 2014-08-22        1
```

1.3 Feature Engineering

In [3]:
```
# Create binary has employee left
dat['has_left'] = dat['quit_date'].notnull()

# See how long the people who have left have been working
dat['days_worked'] = dat['quit_date'] - dat['join_date']

# Separate out join_date information
dat['join_month'] = dat['join_date'].dt.strftime('%B')
dat['join_year'] = dat['join_date'].dt.strftime('%Y')

# Print some example rows of new data
print(dat.head())
```

```
   employee_id  company_id              dept  seniority    salary  join_date  \
0      13021.0           7  customer_service         28   89000.0 2014-03-24
1     825355.0           7         marketing         20  183000.0 2013-04-29
2     927315.0           4         marketing         14  101000.0 2014-10-13
3     662910.0           7  customer_service         20  115000.0 2012-05-14
4     256971.0           2      data_science         23  276000.0 2011-10-17

   quit_date  dept_id  has_left days_worked join_month join_year
0 2015-10-30        0      True    585 days      March      2014
1 2014-04-04        4      True    340 days      April      2013
2        NaT        4     False         NaT    October      2014
3 2013-06-07        0      True    389 days        May      2012
4 2014-08-22        1      True   1040 days    October      2011
```

1.4 Data Summarization

In [4]:
```
print("Overall leave rate: %f" % dat['has_left'].mean())
```

```
Overall leave rate: 0.546919
```

In [5]:
```
# Simple summaries of numeric values
dat.describe()
```

Out[5]:
```
         employee_id   company_id     seniority         salary       dept_id  \
count   24702.000000  24702.000000  24702.000000   24702.000000  24702.000000
mean   501604.403530      3.426969     14.127803  138183.345478      1.955995
std    288909.026101      2.700011      8.089520   76058.184573      1.862562
min        36.000000      1.000000      1.000000   17000.000000      0.000000
25%    250133.750000      1.000000      7.000000   79000.000000      0.000000
50%    500793.000000      2.000000     14.000000  123000.000000      1.000000
75%    753137.250000      5.000000     21.000000  187000.000000      4.000000
max    999969.000000     12.000000     99.000000  408000.000000      5.000000

                   days_worked
count                    13510
mean   613 days 11:41:01.643227
std    328 days 14:56:33.800149
min          102 days 00:00:00
25%          361 days 00:00:00
50%          417 days 00:00:00
75%          781 days 00:00:00
max         1726 days 00:00:00
```

In [6]:
```
# Summarize some information by company
print(dat.groupby('company_id').count())
print()
print(dat.groupby('company_id').mean())
print()
print(dat.groupby('company_id')[['seniority']].describe())
```

```
            employee_id  dept  seniority  salary  join_date  quit_date  \
company_id
1                  8486  8486       8486    8486       8486       4621
2                  4222  4222       4222    4222       4222       2206
3                  2749  2749       2749    2749       2749       1531
4                  2062  2062       2062    2062       2062       1153
5                  1755  1755       1755    1755       1755        983
6                  1291  1291       1291    1291       1291        712
7                  1224  1224       1224    1224       1224        692
8                  1047  1047       1047    1047       1047        579
9                   961   961        961     961        961        529
10                  865   865        865     865        865        480
```

11	16	16	16	16	16	12
12	24	24	24	24	24	12

	dept_id	has_left	days_worked	join_month	join_year
company_id					
1	8486	8486	4621	8486	8486
2	4222	4222	2206	4222	4222
3	2749	2749	1531	2749	2749
4	2062	2062	1153	2062	2062
5	1755	1755	983	1755	1755
6	1291	1291	712	1291	1291
7	1224	1224	692	1224	1224
8	1047	1047	579	1047	1047
9	961	961	529	961	961
10	865	865	480	865	865
11	16	16	12	16	16
12	24	24	12	24	24

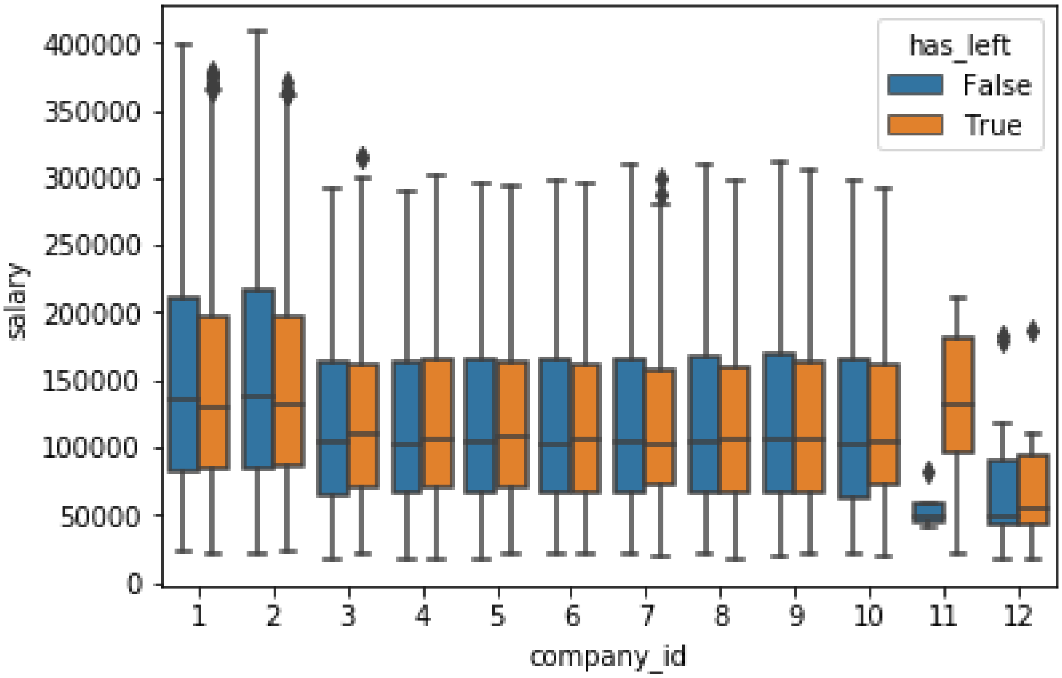
	employee_id	seniority	salary	dept_id	has_left
company_id					
1	501773.268324	14.141999	152167.570115	1.957459	0.544544
2	503864.736618	14.297489	155728.090952	1.949313	0.522501
3	496656.524918	14.054565	122118.588578	1.993452	0.556930
4	513380.616392	14.023763	122721.144520	1.923860	0.559166
5	507257.065527	14.474644	123348.717949	2.026211	0.560114
6	490152.278079	14.089853	119925.639040	1.920991	0.551510
7	501416.076797	13.906046	121582.516340	1.926471	0.565359
8	493358.904489	13.867240	122284.622732	1.957975	0.553009
9	505596.132154	13.778356	123905.306972	1.955255	0.550468
10	490834.589595	14.089017	121553.757225	1.902890	0.554913
11	437283.312500	14.375000	109562.500000	1.750000	0.750000
12	442431.541667	11.166667	73000.000000	1.333333	0.500000

	seniority							
	count	mean	std	min	25%	50%	75%	max
company_id								
1	8486.0	14.141999	8.157523	1.0	7.00	14.0	21.00	99.0
2	4222.0	14.297489	8.024813	1.0	7.00	14.0	21.00	29.0
3	2749.0	14.054565	8.022571	1.0	7.00	14.0	21.00	29.0
4	2062.0	14.023763	8.001208	1.0	7.00	14.0	21.00	29.0
5	1755.0	14.474644	8.067900	1.0	8.00	14.0	21.00	29.0
6	1291.0	14.089853	8.072519	1.0	7.00	14.0	21.00	29.0
7	1224.0	13.906046	7.921435	1.0	7.00	13.0	20.00	29.0
8	1047.0	13.867240	7.952569	1.0	7.00	13.0	20.00	29.0
9	961.0	13.778356	8.224062	1.0	6.00	13.0	21.00	29.0
10	865.0	14.089017	8.449889	1.0	7.00	14.0	20.00	98.0
11	16.0	14.375000	8.585841	1.0	7.25	16.0	20.75	26.0
12	24.0	11.166667	8.036150	1.0	3.75	9.5	17.25	28.0

There are possibly some seniority outliers in company 1 and company 10, with seniorities 99 and 98 respectively.

```
In [7]: # Boxplot of salary for each company, split into groups of employees who remain and ha
        sns.boxplot(x='company_id', y='salary', hue='has_left', data = dat)
```

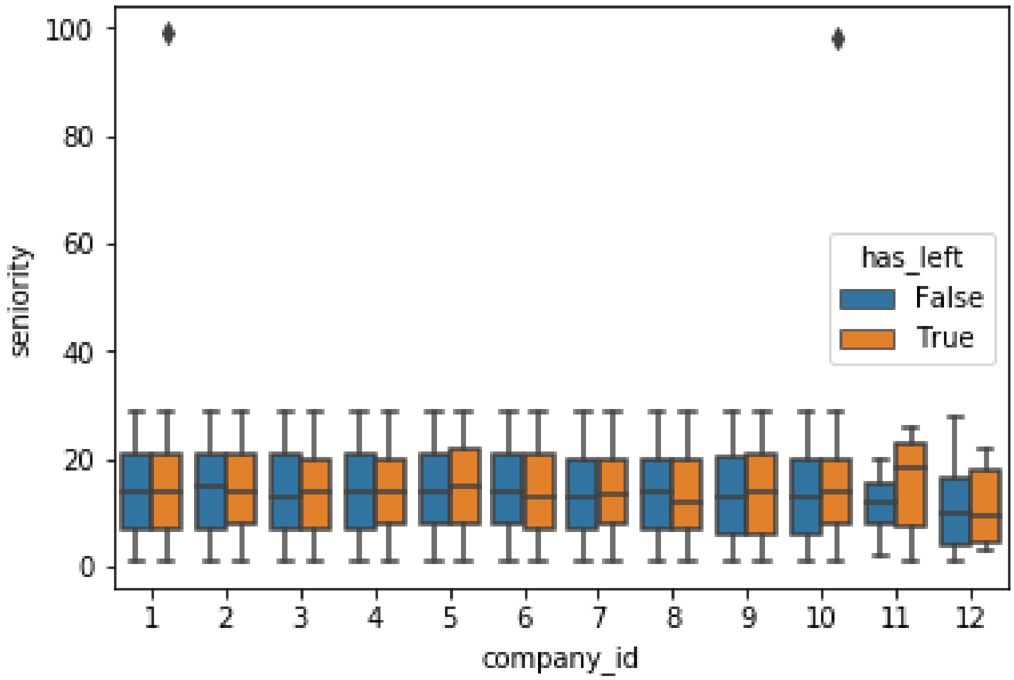
```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7fad76339898>
```



There doesn't seem to be a strong visual trend that the salary distributions differ between those who stay and leave in an individual company.

```
In [8]: # Boxplot of seniority for each company, split into groups of employees who remain and
        sns.boxplot(x='company_id', y='seniority', hue='has_left', data = dat)
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fad6d24db70>
```



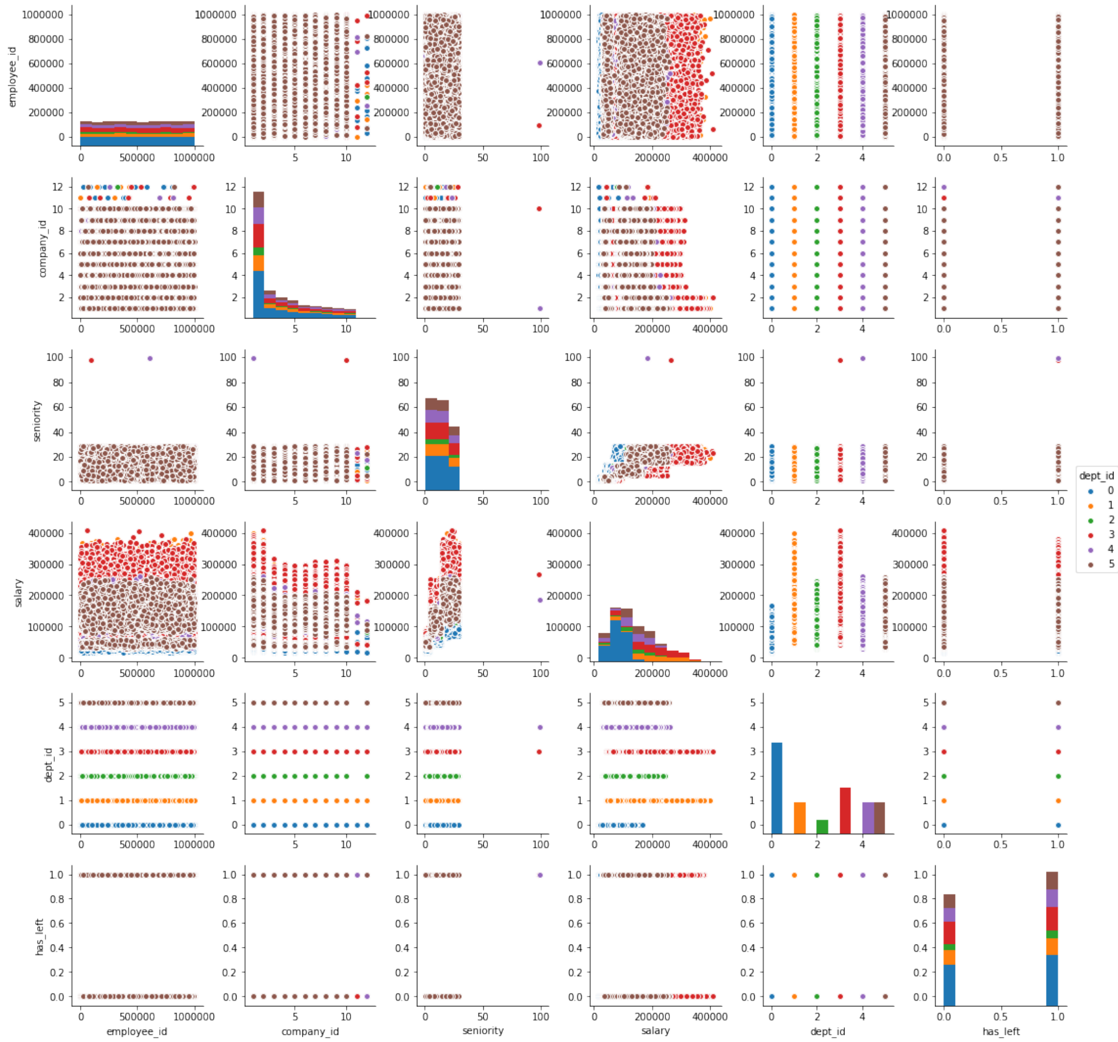
Again, there do not seem to be major differences in seniority between those who have left and those who stay. Note: Here we can clearly see the two seniority outliers previously mentioned.

```
In [9]: # Pair plots across dataset colored by company ID
        i = 9
        print(dat.columns[0:i])
        g = sns.pairplot(dat.iloc[:,0:i], hue='company_id')
```

```
Index(['employee_id', 'company_id', 'dept', 'seniority', 'salary', 'join_date',
       'quit_date', 'dept_id', 'has_left'],
      dtype='object')
```


```
In [10]: # Pair plots across dataset colored by department ID
         i = 9
         print(dat.columns[0:i])
         g = sns.pairplot(dat.iloc[:,0:i], hue='dept_id')
```

```
Index(['employee_id', 'company_id', 'dept', 'seniority', 'salary', 'join_date',
       'quit_date', 'dept_id', 'has_left'],
      dtype='object')
```



1.5 RandomForest Classifier

Here we will fit a RandomForest Classifier to see which factors are important in determining whether an employee stays or leaves a company. Our binary outcome will be whether or not an employee has left.

In [11]:
```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Add one-hot encoding for department and company IDs
df = pd.concat([dat, pd.get_dummies(dat['dept'])], axis = 1)
df = pd.concat([df, pd.get_dummies(dat['company_id'])], axis = 1)

# Add one-hot encoding of join year/month
df = pd.concat([df, pd.get_dummies(df['join_month'])], axis = 1)
```

```
df = pd.concat([df, pd.get_dummies(df['join_year'])], axis = 1)


# Drop some features that we don't wish to include in our regression
drop_feat = ['employee_id', 'company_id', 'dept', 'join_date', 'quit_date', 'has_left
X = df.drop(drop_feat, axis = 1)

# Print out our features
print(X.columns)

# Outcome
y=dat['has_left']

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% traini
```

```
Index([       'seniority',           'salary', 'customer_service',
           'data_science',           'design',         'engineer',
              'marketing',            'sales',                  1,
                        2,                  3,                  4,
                        5,                  6,                  7,
                        8,                  9,                 10,
                       11,                 12,            'April',
                 'August',         'December',         'February',
                'January',             'July',             'June',
                  'March',              'May',         'November',
                'October',        'September',           '2011',
                   '2012',             '2013',             '2014',
                   '2015'],
      dtype='object')
```

In [12]:
```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create classifier using 100 splits
clf = RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
```

In [13]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.767237889624
Precision: 0.765103802193
Recall: 0.820410205103
```

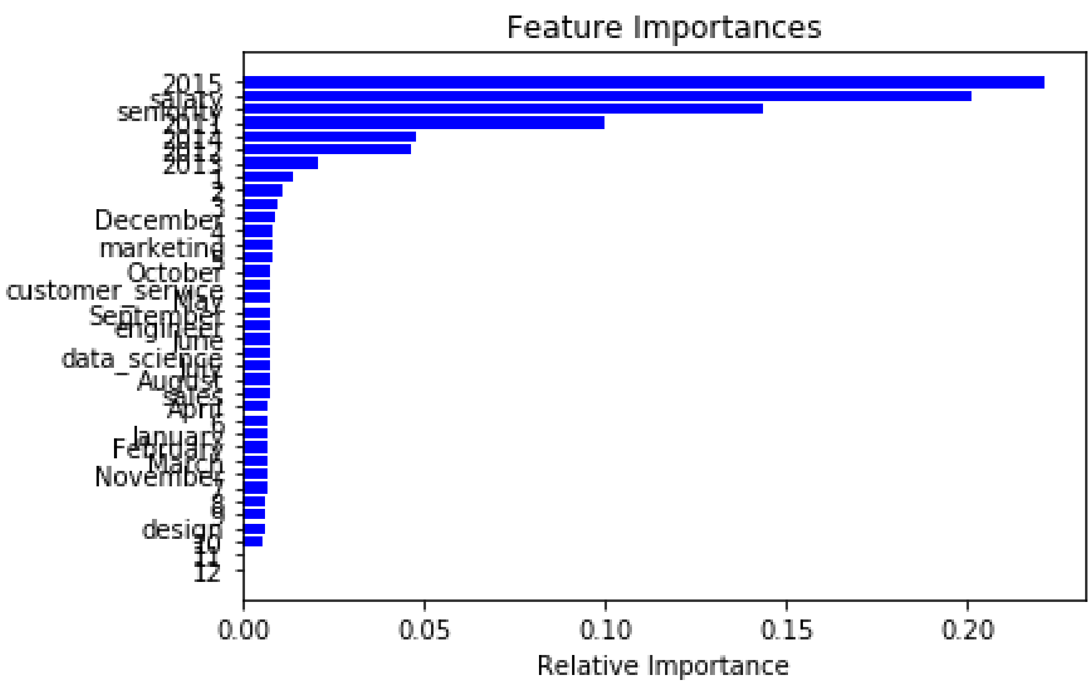
76.8% Accuracy, considering a base-line accuracy would be around 55% if we simply guessed that all employees had left.

In [14]:
```
# Look at which features are important
feature_imp = pd.Series(clf.feature_importances_,index=X.columns).sort_values(ascendi
feature_imp
```

Out[14]:
```
2015                0.221112
salary              0.201357
seniority           0.143729
2011                0.099987
2014                0.047659
2012                0.046551
2013                0.020804
1                   0.013757
2                   0.010988
3                   0.009747
December            0.008963
4                   0.008691
marketing           0.008237
5                   0.008153
October             0.007915
customer_service    0.007867
May                 0.007801
September           0.007739
engineer            0.007705
June                0.007677
data_science        0.007646
July                0.007631
August              0.007496
sales               0.007451
April               0.007053
6                   0.006958
January             0.006955
February            0.006915
March               0.006820
November            0.006769
7                   0.006763
8                   0.006383
9                   0.006329
design              0.006268
```

```
10                    0.005422
11                    0.000371
12                    0.000329
dtype: float64
```

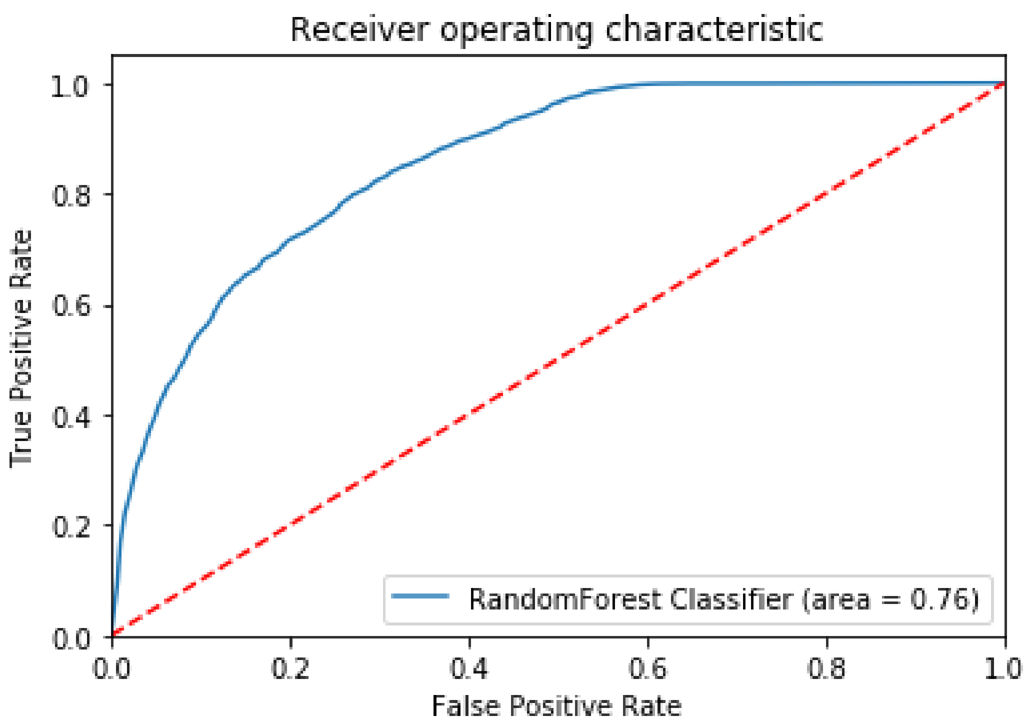
In [15]:
```python
# Plot feature importances
features = X.columns
importances = clf.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



Overall, the important features tend to be when the employee joined the company, where employees who have been at the company for a longer time tend to be more likely to have left the company. This makes sense intuitively and could potentially be seen as a bias (those individuals technically have had more "exposure time" for the event of "leaving their job"). Other important factors included salary and seniority. The company seems to be relatively unimportant as well as the department an individual worked in and particular month they started in.

In [16]:
```python
# ROC-curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
rf_roc_auc = roc_auc_score(y_test, clf.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, clf.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='RandomForest Classifier (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



1.6 Logistic Regression

Trying to get a normal logistic regression to work... but having some trouble because I never get anyone classified as staying... I think this might have something to do with how I'm encoding the categorical variables.

```
In [17]: import statsmodels.api as sm

         X_logistic = X
         y_logistic = dat['has_left']
```

```python
drop_baseline = [1, 'customer_service', '2011', 'January']
X_logistic = X_logistic.drop(drop_baseline, axis = 1)
#X_logistic = X_logistic.iloc[:,:2]

logit_model=sm.Logit(y_logistic,X_logistic)
result=logit_model.fit()
print(X_logistic.columns)
print(result.summary())
print(np.exp(result.params))
```

```
Optimization terminated successfully.
         Current function value: 0.445579
         Iterations 9
Index([   'seniority',       'salary', 'data_science',       'design',
           'engineer',    'marketing',        'sales',              2,
                    3,                4,              5,              6,
                    7,                8,              9,             10,
                   11,               12,        'April',     'August',
           'December',     'February',         'July',       'June',
              'March',          'May',     'November',    'October',
          'September',         '2012',         '2013',       '2014',
               '2015'],
      dtype='object')
                           Logit Regression Results
==============================================================================
Dep. Variable:               has_left   No. Observations:                24702
Model:                          Logit   Df Residuals:                    24669
Method:                           MLE   Df Model:                           32
Date:                Wed, 20 Feb 2019   Pseudo R-squ.:                  0.3530
Time:                        13:24:34   Log-Likelihood:                -11007.
converged:                       True   LL-Null:                       -17013.
                                        LLR p-value:                     0.000
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
seniority        0.0291      0.003      8.692      0.000       0.023       0.036
salary       -1.461e-07   4.87e-07     -0.300      0.764    -1.1e-06    8.08e-07
data_science     0.1127      0.080      1.403      0.160      -0.045       0.270
design           0.3664      0.081      4.547      0.000       0.208       0.524
engineer         0.0648      0.076      0.850      0.395      -0.085       0.214
marketing        0.4187      0.061      6.879      0.000       0.299       0.538
sales            0.4300      0.060      7.116      0.000       0.312       0.548
2                0.2782      0.048      5.808      0.000       0.184       0.372
3                0.4883      0.059      8.297      0.000       0.373       0.604
4                0.4922      0.065      7.607      0.000       0.365       0.619
5                0.4505      0.070      6.432      0.000       0.313       0.588
6                0.4000      0.079      5.060      0.000       0.245       0.555
```

```
7                  0.4237      0.080      5.298      0.000       0.267       0.580
8                  0.4537      0.087      5.232      0.000       0.284       0.624
9                  0.4682      0.091      5.171      0.000       0.291       0.646
10                 0.5003      0.095      5.258      0.000       0.314       0.687
11                 0.8999      0.645      1.394      0.163      -0.365       2.165
12                -0.0611      0.501     -0.122      0.903      -1.043       0.920
April              1.3040      0.069     18.805      0.000       1.168       1.440
August             1.0821      0.070     15.424      0.000       0.945       1.220
December           0.4299      0.066      6.525      0.000       0.301       0.559
February           1.3409      0.074     18.203      0.000       1.196       1.485
July               1.0943      0.069     15.754      0.000       0.958       1.230
June               1.1905      0.070     16.906      0.000       1.053       1.329
March              1.3080      0.072     18.273      0.000       1.168       1.448
May                1.2385      0.070     17.674      0.000       1.101       1.376
November           0.6923      0.069      9.971      0.000       0.556       0.828
October            0.8407      0.067     12.576      0.000       0.710       0.972
September          0.9440      0.068     13.806      0.000       0.810       1.078
2012              -0.3346      0.049     -6.842      0.000      -0.430      -0.239
2013              -1.3032      0.047    -27.775      0.000      -1.395      -1.211
2014              -2.3751      0.049    -48.917      0.000      -2.470      -2.280
2015              -6.7505      0.168    -40.168      0.000      -7.080      -6.421
================================================================================
seniority       1.029510
salary          1.000000
data_science    1.119263
design          1.442559
engineer        1.066958
marketing       1.519982
sales           1.537287
2               1.320720
3               1.629499
4               1.635849
5               1.569140
6               1.491770
7               1.527550
8               1.574134
9               1.597125
10              1.649295
11              2.459475
12              0.940704
April           3.684119
August          2.950772
December        1.537125
February        3.822322
July            2.987130
June            3.288864
March           3.698587
May             3.450285
```

```
November        1.998239
October         2.318092
September       2.570320
2012            0.715600
2013            0.271654
2014            0.093009
2015            0.001170
dtype: float64
```

```
/home/jeremy/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarni
  from pandas.core import datetools
```

```
In [18]: from sklearn.linear_model import LogisticRegression
         from sklearn import metrics
         X_train, X_test, y_Tryingtrain, y_test = train_test_split(X_logistic, y_logistic, test
         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)
         print(X_train.shape)
```

```
(17291, 33)
```

```
In [19]: y_pred = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.s
```

```
Accuracy of logistic regression classifier on test set: 0.55
```

```
In [20]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred)
         print(confusion_matrix)
```

```
[[   0 3352]
 [   0 4059]]
```

```
In [21]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

      False       0.00      0.00      0.00      3352
       True       0.55      1.00      0.71      4059

avg / total       0.30      0.55      0.39      7411
```

```
/home/jeremy/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Unde
  'precision', 'predicted', average, warn_for)
```

```
In [22]: from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve
         logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
         fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
         plt.figure()
         plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.05])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver operating characteristic')
         plt.legend(loc="lower right")
         plt.savefig('Log_ROC')
         plt.show()
```