In [3]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from datetime import datetime, timedelta
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
df = pd.read_csv('employee_retention_data.csv')
last_date = datetime(2015,12,13)
```

# Data Preparation:

1) The goal is to predict employee retention and its main drivers.

2) The data includes the record of each employee in a row. From the "quit_date" column I can determine whether an employee has quit or not. Those with a date in "quit_date" column will have 1 as an indication of class of those who quit their jobs, and zero otherwise.

3) The remaining columns ['employee_id', 'company_id', 'dept', 'seniority', 'salary', 'join_date'] carry information about employees. The goal is to use information provided in these columns to find whether an employee quits, and also what part of information indicated by these columns can perform as a good predictor. in the last section I answer the question, how does each predictor impact the employee retention.

4) Since join_date is not a integer of categorical factor, I exploit a new feature called 'days_since_lastdate' to be able to use the information provided by 'join_date' column. as the question states the last day to analyze this data is 2015-12-13. the difference between this date (last_date= 2015-12-13) and each employees join_date is the number of days since an employee has joined the company. Feauture 'days_since_lastdate' can be used in a numeric format.

5) Since the number of records in the dataset and the number if employee_id's are thesame, and also eachemployee id is a unique number that does not provide information regarding an employees work or characteristics (otherwise the question would stated) therefore employee_id is taken out of the dataset. However, in the plots we can see that there is no association between the employee_id and other features or employee_id and whether an employee guits indicated churn column

6) 'dept' and 'company_id' are categorical data and need to be prepared to be fed to a classifier model. I am making a binary variables for each instance of the two features.

In [5]:
```
print( 'Number of records in dataset = '+ str(len(df)) +
      ' which is equal to the number of unique employee ids = '+ str(len(df['e
mployee_id'].unique())))
```

```
Number of records in dataset = 24702 which is equal to the number of unique e
mployee ids = 24702
```

In [6]:
```
emp_out_idx = df[df['quit_date'].isnull()==False].index
emp_in_idx = df[df['quit_date'].isnull()==True].index

df['churn'] = 0
df.loc[emp_out_idx, 'churn'] = 1
df['join_date'] = pd.to_datetime(df['join_date'],errors='raise')
df['quit_date'] = pd.to_datetime(df['quit_date'],errors='raise')


df['days_since_lastdate'] = last_date  - df['join_date']
df['days_since_lastdate'] = df['days_since_lastdate']/timedelta(days = 1)
```

In [7]:
```
df.head()
```

Out[7]:

| | employee_id | company_id | dept | seniority | salary | join_date | quit_date |
|---|---|---|---|---|---|---|---|
| **0** | 13021.0 | 7 | customer_service | 28 | 89000.0 | 2014-03-24 | 2015-10-30 |
| **1** | 825355.0 | 7 | marketing | 20 | 183000.0 | 2013-04-29 | 2014-04-04 |
| **2** | 927315.0 | 4 | marketing | 14 | 101000.0 | 2014-10-13 | NaT |
| **3** | 662910.0 | 7 | customer_service | 20 | 115000.0 | 2012-05-14 | 2013-06-07 |
| **4** | 256971.0 | 2 | data_science | 23 | 276000.0 | 2011-10-17 | 2014-08-22 |

# Data Analysis:

In order to choose the best model for learning from data I need to: 1) whether there is an extereme class imblance in data. The histogram (Density Distribution of Churn) shows that the class imbalance is acceptable as each class includes around 50% of data 2) investigate whetheher there is any association at all between the features and churn 3) investigate whether there is any redundancy in the information provided by features; this includes linear relationship between features if they are continous, or

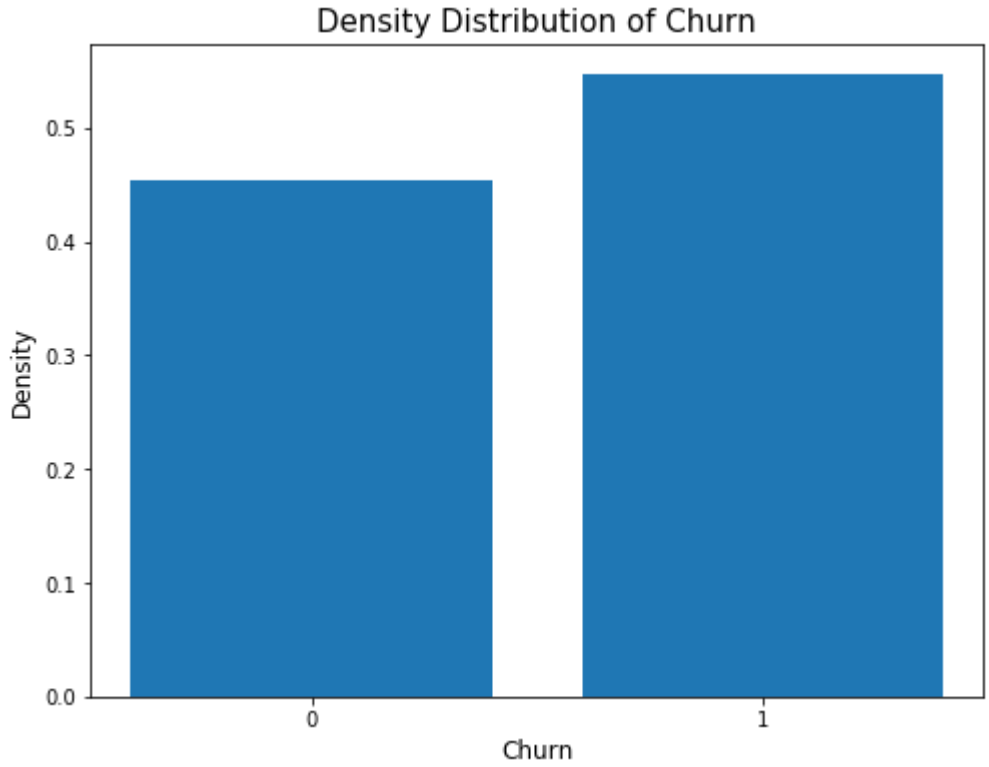
Observation: the department is a categorical data that does not have an ascending or descending order. the bar chart shows whethe we can find a difference between department of those who have quit with those who staye.

Observation: scatter plots do not show an association between employee id/company id as featueres and the churn as the response variable. I mentioned that employe_id will be excluded from models previously.

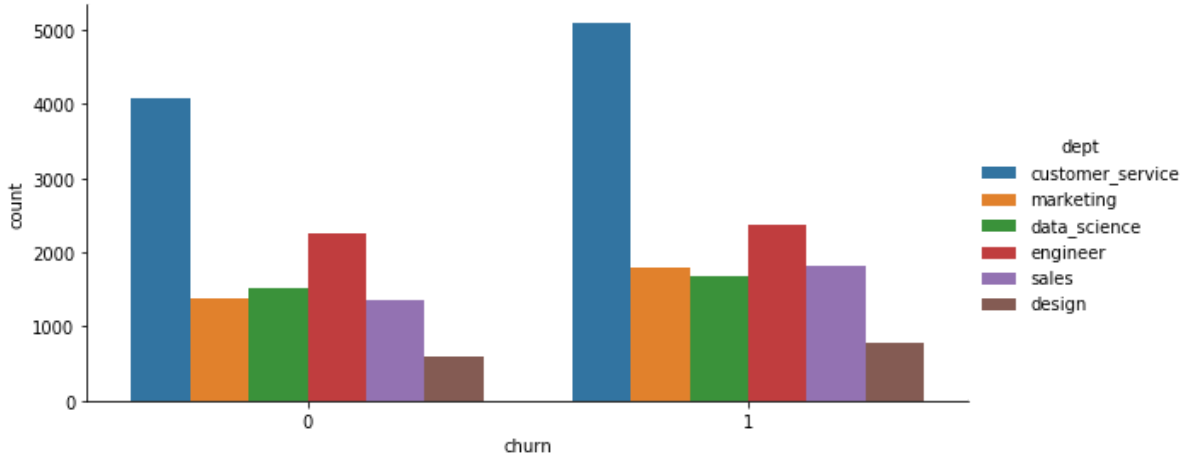
Observation: company id have some relationship with dept when looking at the employees' salary. Therefore there might be a conditional relationship. Therefore, I will keep the company_id in the feautures and use a feature selection method (regularization) to exclude redunadancy and prevent high variance in the estimators.

Observation: seniority and salary are correlated and we expect to see that models that do feature selection exclude of of the two.

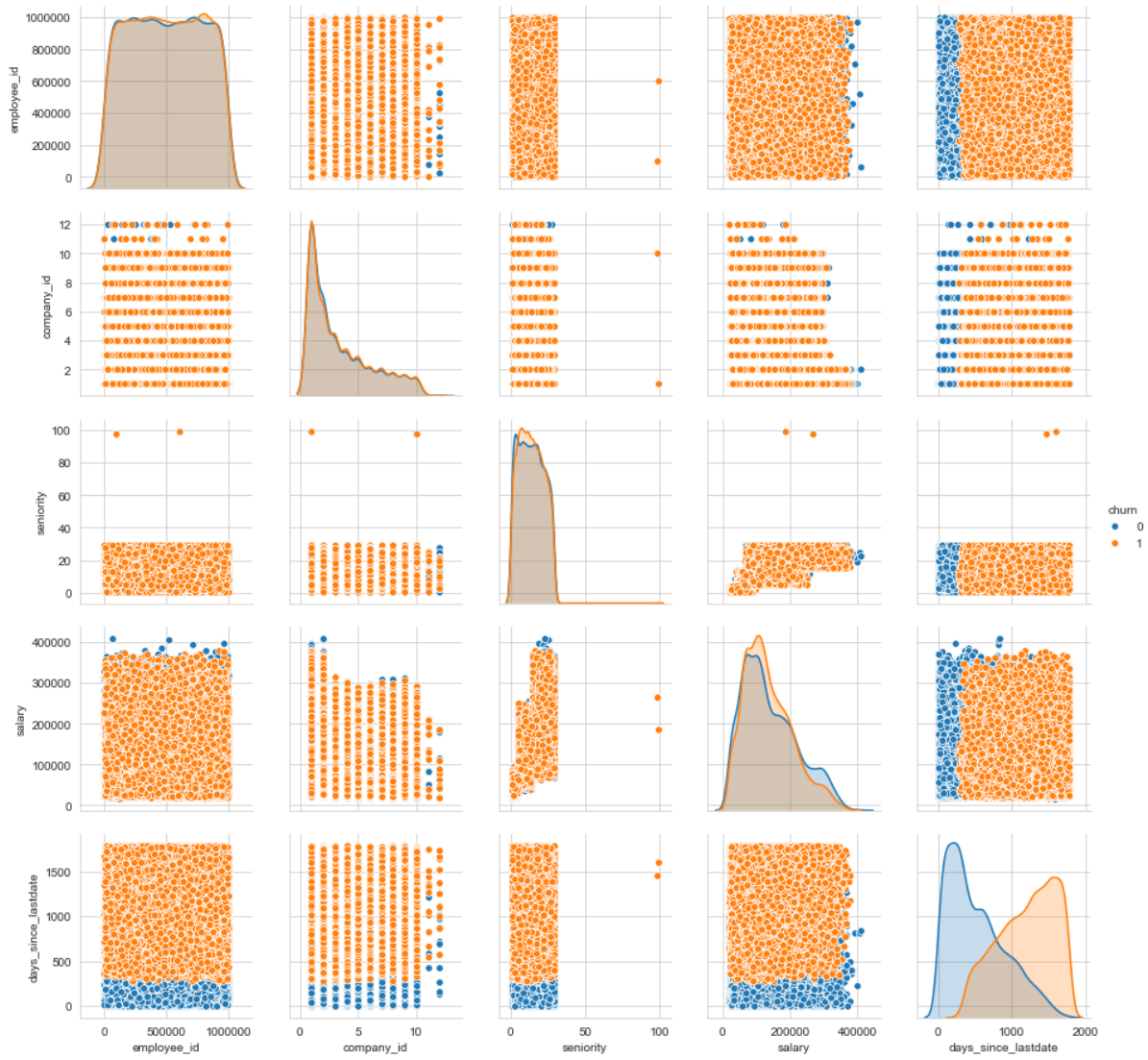
In [8]:
```
plt.figure(figsize = (8,6))
plt.bar(df['churn'].unique(),[len(emp_out_idx)/len(df), len(emp_in_idx)/len(df
)] )
plt.xlabel('Churn', fontsize = 12)
plt.xticks([1,0])
plt.ylabel('Density', fontsize = 12)
plt.title('Density Distribution of Churn', fontsize = 15)
plt.show()
```



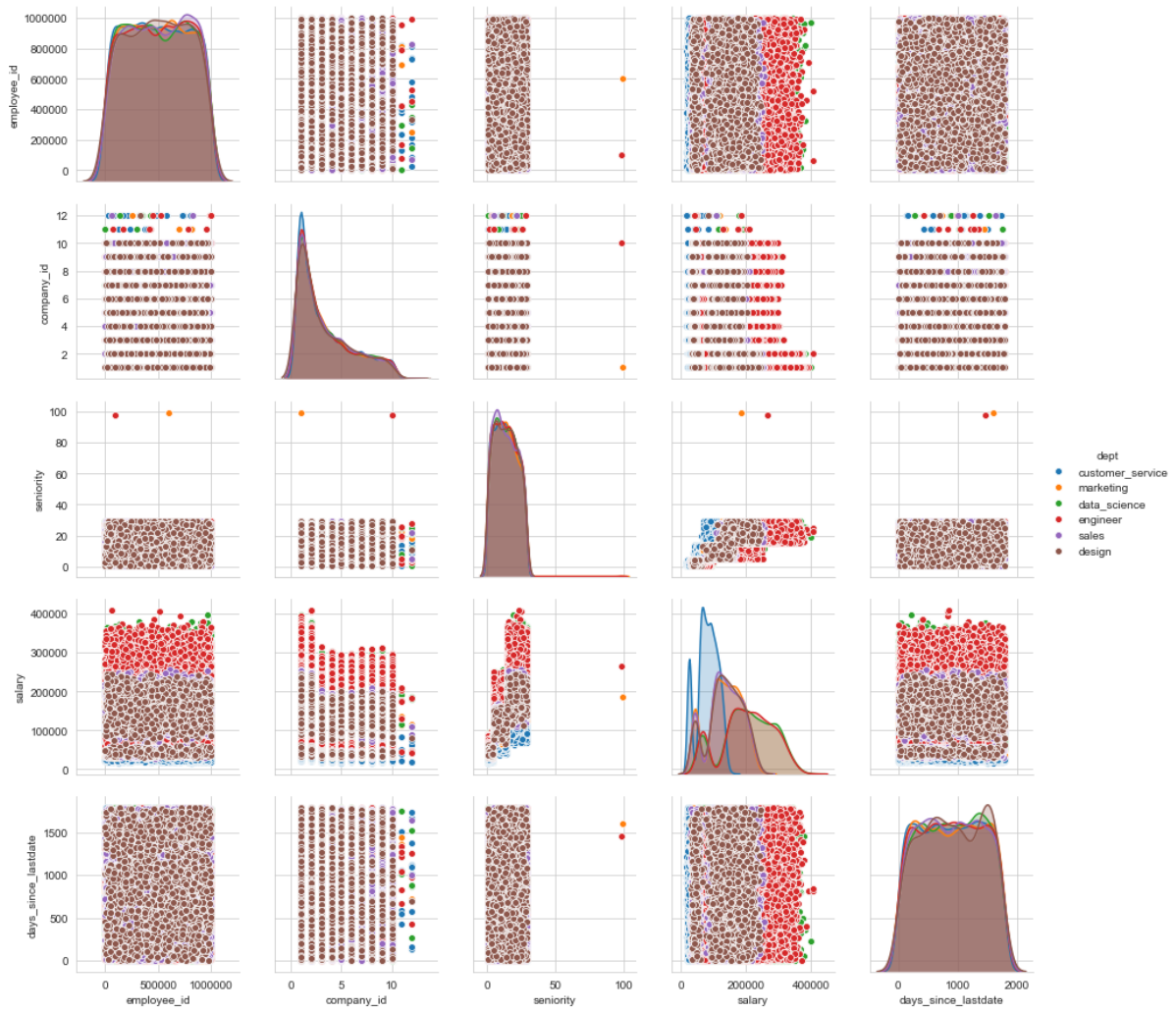

In [9]:
```
df['churn'] = df['churn'].astype(str)
g = sns.catplot(x="churn", hue="dept", data=df, kind="count", height=4, aspect
=2);
```



In [10]:
```
sns.set_style("whitegrid")
sns.pairplot(df, hue="churn",vars=['employee_id', 'company_id', 'seniority',
'salary','days_since_lastdate']);
```



In [11]:
```
sns.set_style("whitegrid")
sns.pairplot(df, hue="dept",vars=['employee_id', 'company_id', 'seniority', 's
alary','days_since_lastdate']);
```



# Model Selection:

Due to having 3 non-categorical (integer and continous) features and 2 categorical features with a binary response variable, I used logistic regression as the first model and used the information at each step to build a better model. After turning 'company_id' and 'dept' into binary variables, I did the following:

1) In order to identify the best features to includein the model, I used regularization as means which is a non-subjective technique that selects the best set of features by penelilising those that are multicolinear, redundent or do not provide predictive power. therefore, I normalized data before splitting into test and train due to using regularization. Regularization also helps overfitting which will futhur be investigated.

2) I used crossvalidation on the train posrtion of data to better investigate the performance of models

3) Models are tuned to have better TEST_ERROR by metrics that measure the error in the portion of data that is unseen by the models to prevent overfitting.

# Metrics

The metrics selected for minimizing the test score is Accuracy, Area under the ROC curve (AUC), and I also look at the precision-recall trade off (F1 score is used as a measure for representing the trade off) with confusion matrix.

- In this problem although predicting those who will churn may have greater value, but since the data does not have class imbalance, accuracy along with AUC thouroghly explains goodness of fit.
- We care about identifying those that do not quit, however if we cared as much about them as we care about those who quit, we would also include specifity. But those who quit are those who cause more cost for the company.

# Finding Main Factors of Churn:

For identifying best drivers of retention, coefficients generated by the Ridge model (L2) is used for this purose and interpretation; since coefficients do not fluctuate on small data changes as is the case with unregularized or L1 model.

The reason is that when using unregularized model, features can be correlated and therefore the variance of coefficients may be amplified. This means if a slightly different data set was used to train the model, there could be large variations in coefficients. Similarly, due to the nature of Lasso, small data changes can cause therefore, the resulting set of active features may change.

In [12]:
```
feature_cols = ['company_id', 'dept', 'seniority', 'salary', 'days_since_lastd
ate']
data = df[feature_cols]
```

In [13]:
```
cat_vars=['company_id','dept']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1
cat_vars=['company_id','dept']
data_vars=data.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data=data[to_keep]
```

In [14]:
```
cols_to_transform = ['seniority','salary','days_since_lastdate']
data['seniority'] = (data['seniority'] - data['seniority'].min())/(data['senio
rity'].max() - data['seniority'].min())
data['salary'] = (data['salary'] - data['salary'].min())/(data['salary'].max()
- data['salary'].min())
data['days_since_lastdate'] = (data['days_since_lastdate'] - data['days_since_
lastdate'].min())/(data['days_since_lastdate'].max() - data['seniority'].min
())
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(data, df['churn'].astype(i
nt), test_size=0.33, random_state=42)
```

In [16]:
```
logmodel = LogisticRegression().fit(X_train, y_train)
predicted_classes = logmodel.predict(X_test)
predicted_probs = logmodel.predict_proba(X_test)
accuracy = accuracy_score(y_test,predicted_classes)
parameters = logmodel.coef_
parameters, accuracy
```

Out[16]:
```
(array([[ 1.192871  , -0.98662483,  6.73029734, -0.26163096, -0.36005655,
         -0.29656556, -0.18068315, -0.20599129, -0.28885287, -0.2070604 ,
         -0.25320612, -0.21105753, -0.17150078,  0.38688699, -0.18917659,
         -0.43424446, -0.44782931, -0.37309782, -0.47922155, -0.2115054 ,
         -0.29299627]]), 0.7872914622178606)
```

In [17]:
```
# Confusion matrix
pd.crosstab(y_test, predicted_classes, rownames=['Actual'], colnames=['Predict
ed'])
```

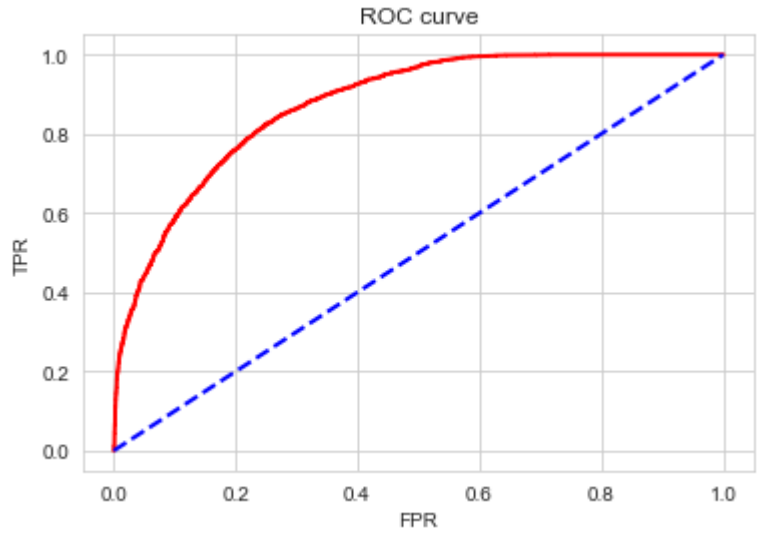
Out[17]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2811 | 860 |
| 1 | 874 | 3607 |

In [18]:
```
precision_recall_fscore_support(y_test, predicted_classes, average='macro')
```

Out[18]:
```
(0.7851496531628444, 0.7853428298093996, 0.7852438705123166, None)
```

In [19]:
```
prob_retention =[]
for i in range(len(predicted_probs)):
    prob_retention.append(predicted_probs[i][1])
##Computing false and true positive rates
fpr, tpr, thresholds = metrics.roc_curve(y_test, prob_retention)
plt.figure()
##Adding the ROC
plt.plot(fpr, tpr, color='red',lw=2, label='ROC curve')
##Random FPR and TPR
plt.plot([0, 1], [0, 1], color='blue', lw=2, linestyle='--')
##Title and label
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()
```

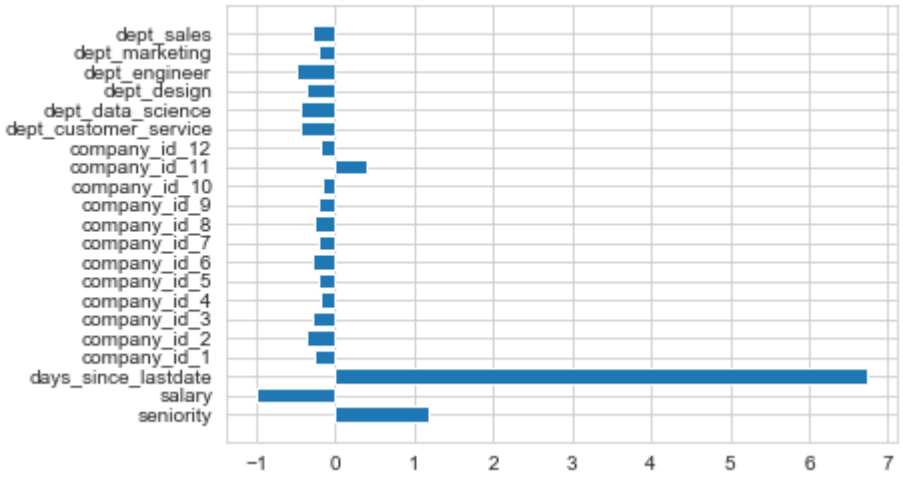



In [20]:
```
auc = metrics.roc_auc_score(y_test, prob_retention)
auc
```

Out[20]: 0.8743011368378767

In [21]:
```
plt.barh(X_train.columns, parameters[0])
```

Out[21]:
```
<BarContainer object of 21 artists>
```




# Improving Models:

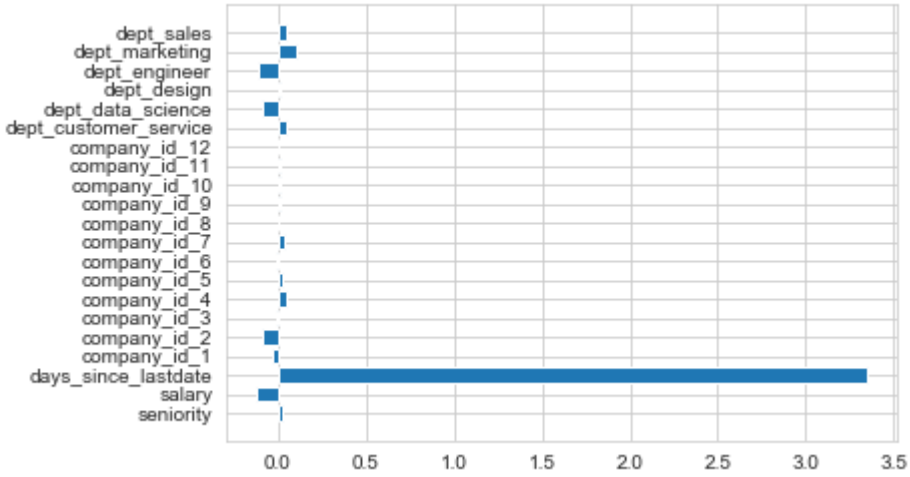
1) Regularization: in order to find the balance in bias-variance tradeoff, I use regularization as measn to decrease variance of coefficients (the estimations) and therefore allowing bias to increase,which will result in prevention of overfitting.

Through L1 lasso regularization, I find the main drivers to include in the model and through L2, Ridge, i find the direction that each predictor affects the churn when all features are maintained. this helps identifying features that are inter-correlated when compared to direction in Lasso.

2) Cross Validation: for model selection, using CV avoids chosing models that overfit and help select models that adjust performance measures for data that is not seen by the model. therefore, it helps finding the predictive model that best represents data.

In [22]:
```
logmodel_cv = LogisticRegressionCV(cv=5, random_state=0,multi_class='ovr').fit
(X_train, y_train)
predicted_classes_cv = logmodel_cv.predict(X_test)
predicted_prob_cv = logmodel_cv.predict_proba(X_test)
accuracy_cv = accuracy_score(y_test,predicted_classes_cv)
parameters_cv = logmodel_cv.coef_
plt.barh(X_train.columns, parameters_cv[0])
```

Out[22]: <BarContainer object of 21 artists>




In [23]:
```
# Confusion matrix
pd.crosstab(y_test, predicted_classes_cv, rownames=['Actual'], colnames=['Pred
icted'])
```

Out[23]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2743 | 928 |
| 1 | 788 | 3693 |

In [24]:
```
prob_retention_cv =[]
for i in range(len(predicted_prob_cv)):
    prob_retention_cv.append(predicted_prob_cv[i][1])

prfs = precision_recall_fscore_support(y_test, predicted_classes_cv, average=
'macro')
auc = metrics.roc_auc_score(y_test, prob_retention_cv)
print ('Logistic Regression Model with CV: has accuracy of ' +str(accuracy_cv)
+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
```

```
Logistic Regression Model with CV: has accuracy of 0.7894995093228655, precis
ion of 0.7880057126568887, recall of 0.7856771205837705, and AUC of 0.8739743
841715295
```

# Lasso Regularization

In order to get a better estimation on what are the main drivers of rentention Lasso (L1) is used.

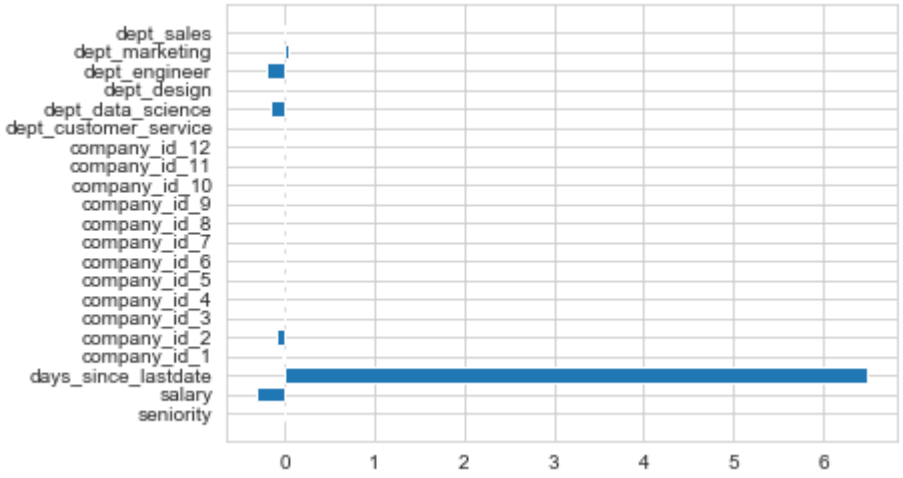
Cross validation helps for getting a accuracy measure without overfifiing.

Result: Notice that comparing the confusion matrix of lasso requlrized model compared to the unregular logistic model shows greater power of predicting those who actually quit, it is also shown in the recall.

- as expected we see that Lasso model zeros out the seniority as it was correlated with the salary.

In [25]:
```
logmodel_cv_l1 = LogisticRegressionCV(cv=5, random_state=0,multi_class='ovr',p
enalty='l1',solver = 'liblinear').fit(X_train, y_train)
predicted_classes_cv_l1 = logmodel_cv_l1.predict(X_test)
predicted_prob_cv_l1 = logmodel_cv_l1.predict_proba(X_test)
accuracy_cv_l1 = accuracy_score(y_test,predicted_classes_cv_l1)
parameters_cv_l1 = logmodel_cv_l1.coef_
plt.barh(X_train.columns, parameters_cv_l1[0])
```

Out[25]: <BarContainer object of 21 artists>




In [26]:
```
# Confusion matrix
pd.crosstab(y_test, predicted_classes_cv_l1, rownames=['Actual'], colnames=['P
redicted'])
```

Out[26]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2803 | 868 |
| 1 | 870 | 3611 |

In [27]:
```
prob_retention_cv_l1 =[]
for i in range(len(predicted_prob_cv_l1)):
    prob_retention_cv_l1.append(predicted_prob_cv_l1[i][1])

pd.crosstab(y_test, predicted_classes_cv_l1, rownames=['Actual'], colnames=['P
redicted'])
prfs= precision_recall_fscore_support(y_test, predicted_classes_cv_l1, average
='macro')
log_prfs = prfs
auc = metrics.roc_auc_score(y_test, prob_retention_cv_l1)
print ('Logistic Regression Model with CV and Lasso: has accuracy of ' +str(ac
curacy_cv_l1)+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
```

```
Logistic Regression Model with CV and Lasso: has accuracy of 0.78680078508341
51, precision of 0.7846715716693937, recall of 0.784699537397253, and AUC of
0.8743440554206565
```

# Ridge Regression

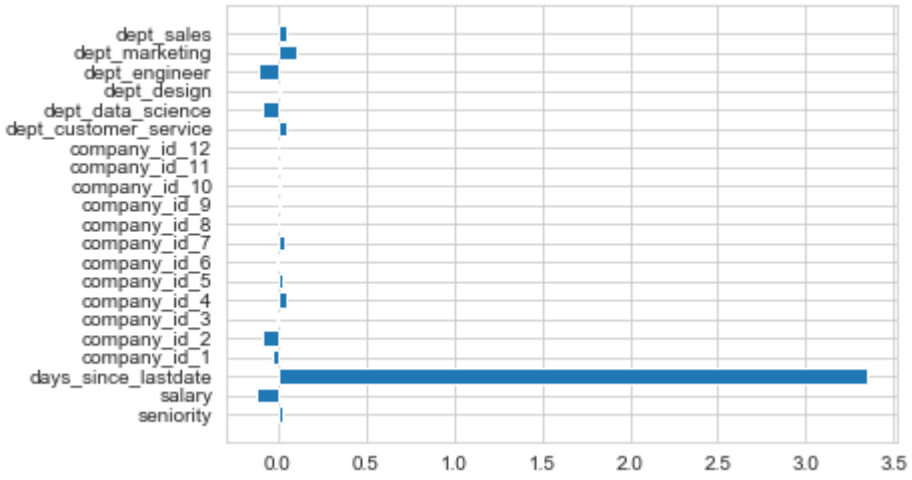
Shrinking features without allowing them to adopt zero helps finding the direction of each feature when all features simultaneously exist in the model.

Although both Lasso and Ridge models provide unique solutions, the coefficients from Ridge regression are stable if slightly different dataset was used to . train the model. as a results coefficients are reliable and can be interpreted for finding drives of retention.

In [28]:
```
logmodel_cv_l2 = LogisticRegressionCV(cv=100, random_state=0,multi_class='ovr'
,penalty='l2').fit(X_train, y_train)
predicted_classes_cv_l2 = logmodel_cv_l2.predict(X_test)
predicted_prob_cv_l2 = logmodel_cv_l2.predict_proba(X_test)
accuracy_cv_l2 = accuracy_score(y_test,predicted_classes_cv_l2)
parameters_cv_l2 = logmodel_cv_l2.coef_
parameters_cv_l2_df = pd.DataFrame(logmodel_cv_l2.coef_)
predicted_probs_cv_l2 = logmodel_cv_l2.predict_proba(X_test)
```

In [29]:
```
plt.barh(X_train.columns, parameters_cv_l2[0])
```

Out[29]: <BarContainer object of 21 artists>




In [30]:
```
pd.crosstab(y_test, predicted_classes_cv_l1, rownames=['Actual'], colnames=['P
redicted'])
```

Out[30]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2803 | 868 |
| 1 | 870 | 3611 |

In [31]:
```
prob_retention_cv_l2 =[]
for i in range(len(predicted_prob_cv_l2)):
    prob_retention_cv_l2.append(predicted_prob_cv_l2[i][1])

pd.crosstab(y_test, predicted_classes_cv_l2, rownames=['Actual'], colnames=['P
redicted'])
prfs= precision_recall_fscore_support(y_test, predicted_classes_cv_l2, average
='macro')
auc = metrics.roc_auc_score(y_test, prob_retention_cv_l2)
print ('Logistic Regression Model with CV and Ridge: has accuracy of ' +str(ac
curacy_cv_l2)+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
```

```
Logistic Regression Model with CV and Ridge: has accuracy of 0.78949950932286
55, precision of 0.7880057126568887, recall of 0.7856771205837705, and AUC of
0.8739727428092983
```

In [32]:
```
logmodel_cv_l2.score(X_train, y_train), logmodel_cv_l2.score(X_test, y_test)
```

Out[32]: (0.7954078549848943, 0.7894995093228655)

# Do we need more sofisticated models:

Why:

1) After turning the categorical features to binary variables, now most of the features are binary. A tree-based model can perform well when the number of categorical features increase due to the constant-point splits.

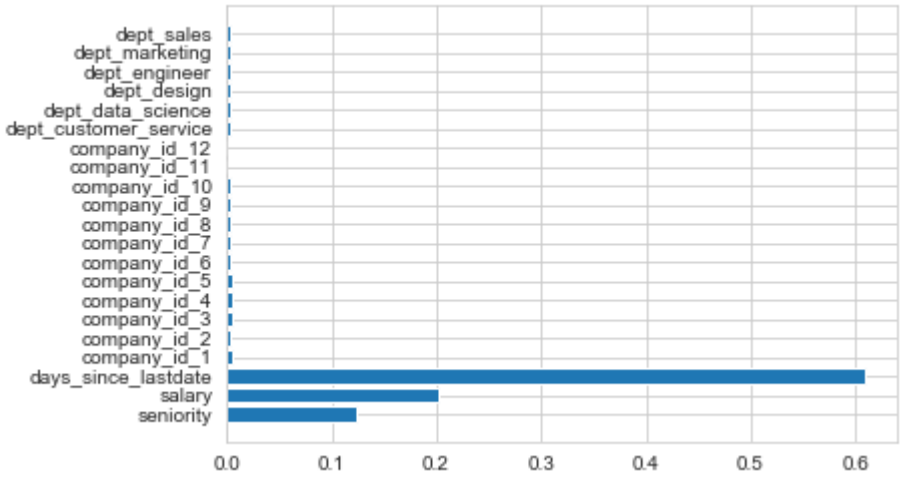
2) A random forest model with 20 trees can promis an accuracy of 99% on TEST data.

In [34]:
```
# 10 trees:
rf = RandomForestClassifier(n_estimators = 10, random_state = 42,criterion='en
tropy')
rf.fit(X_train, y_train)
rf_importance10  = rf.feature_importances_
predicted_classes_rf = rf.predict(X_test)
predicted_prob_rf = rf.predict_proba(X_test)
plt.barh(X_train.columns, rf_importance10)

accuracy_rf = accuracy_score(y_test,predicted_classes_rf)

prob_retention_rf =[]
for i in range(len(predicted_prob_rf)):
    prob_retention_rf.append(predicted_prob_rf[i][1])
prfs= precision_recall_fscore_support(y_test, predicted_classes_rf, average='m
acro')
auc = metrics.roc_auc_score(y_test, prob_retention_rf)
print ('Logistic Regression Model with CV and Ridge: has accuracy of ' +str(ac
curacy_rf)+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
pd.crosstab(y_test, predicted_classes_rf, rownames=['Actual'], colnames=['Pred
icted'])
```

```
Logistic Regression Model with CV and Ridge: has accuracy of 0.75711481844946
03, precision of 0.7547824491156814, recall of 0.7557270015819693, and AUC of
0.838321808032231
```

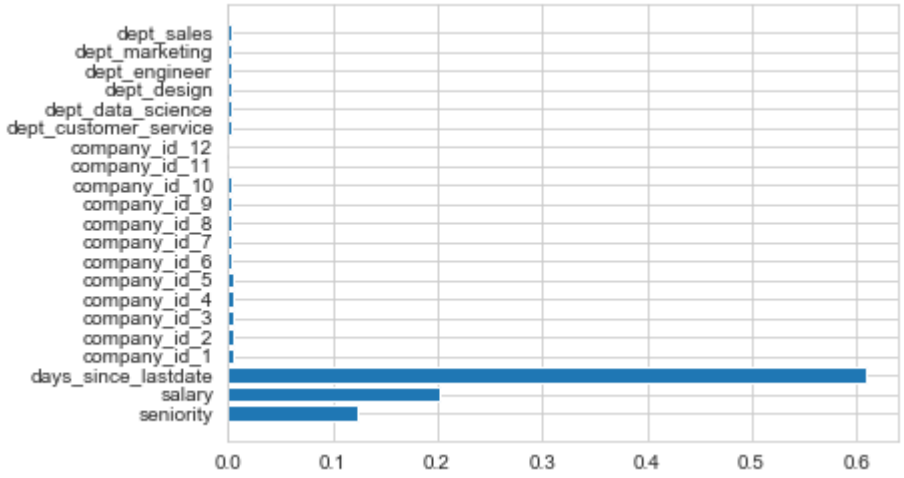
Out[34]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2723 | 948 |
| 1 | 1032 | 3449 |



In [38]:
```
# 50 trees:
rf = RandomForestClassifier(n_estimators = 50, random_state = 42,criterion='en
tropy')
rf.fit(X_train, y_train)
predicted_classes_rf = rf.predict(X_test)
predicted_prob_rf = rf.predict_proba(X_test)
rf_importance = rf.feature_importances_
accuracy_rf = accuracy_score(y_test,predicted_classes_rf)


plt.barh(X_train.columns, rf_importance)


prob_retention_rf =[]
for i in range(len(predicted_prob_rf)):
    prob_retention_rf.append(predicted_prob_rf[i][1])
prfs= precision_recall_fscore_support(y_test, predicted_classes_rf, average='m
acro')
auc = metrics.roc_auc_score(y_test, prob_retention_rf)
print ('Logistic Regression Model with CV and Ridge: has accuracy of ' +str(ac
curacy_rf)+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
```

```
Logistic Regression Model with CV and Ridge: has accuracy of 0.77073110893032
38, precision of 0.7692083606482774, recall of 0.7661183746793492, and AUC of
0.8554524016807306
```




In [39]:
```
pd.crosstab(y_test, predicted_classes_rf, rownames=['Actual'], colnames=['Pred
icted'])
```

Out[39]:

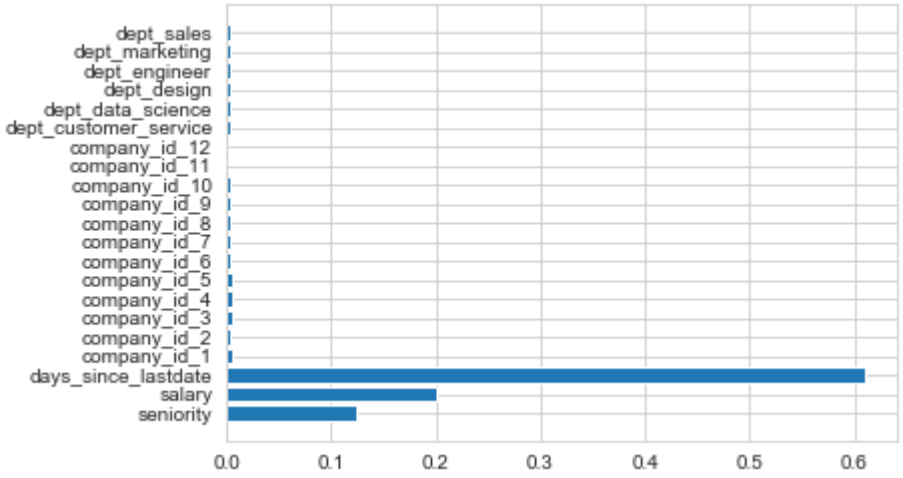
| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2642 | 1029 |
| 1 | 840 | 3641 |

In [37]:
```
# 20 trees:
rf = RandomForestClassifier(n_estimators = 20, random_state = 42,criterion='en
tropy')
rf.fit(X_train, y_train)
rf_importance20  = rf.feature_importances_
predicted_classes_rf = rf.predict(X_test)
predicted_prob_rf = rf.predict_proba(X_test)
plt.barh(X_train.columns, rf_importance20)

accuracy_rf = accuracy_score(y_test,predicted_classes_rf)


prob_retention_rf =[]
for i in range(len(predicted_prob_rf)):
    prob_retention_rf.append(predicted_prob_rf[i][1])
prfs= precision_recall_fscore_support(y_test, predicted_classes_rf, average='m
acro')
auc = metrics.roc_auc_score(y_test, prob_retention_rf)
print ('Logistic Regression Model with CV and Ridge: has accuracy of ' +str(ac
curacy_rf)+
       ', precision of ' +str(prfs[0])+ ', recall of ' +str(prfs[1])+ ', and A
UC of '+str(auc) )
pd.crosstab(y_test, predicted_classes_rf, rownames=['Actual'], colnames=['Pred
icted'])
```

```
Logistic Regression Model with CV and Ridge: has accuracy of 0.76962708537782
14, precision of 0.7674961068007319, recall of 0.7663697766610571, and AUC of
0.8497993677837432
```

Out[37]:

| Predicted | 0 | 1 |
|---|---|---|
| Actual | | |
| 0 | 2693 | 978 |
| 1 | 900 | 3581 |



# Random Forest:

As the number of trees is increased in randomforest the fonal performance measures do not increase more that the slected model with 50 trees.

# Comparison of Random Forest and Logistic Regression with L1 and CV

1) From below comparison between accurcy, AUC and F1- score we see that logistic regression with regularization out performs Randomforest. Randomforest overfits the data as its training accuracy is 99% but logistic model has training accuracy of 79% and test accuracy of 78% with F1 score of 78% (this also confirms that accuracy is a good measure to indicate goodness of fit for this data and model) and AUC of 87% which is greater than its correspondence in random forest. another indication for not having overfitting and having a good biad-variance tradeoff is that, the test accuracy and training accuracy of Logistic regression model are almost equal. whereas this is not the case for randomforest, indicating that random forest overfitts the data.

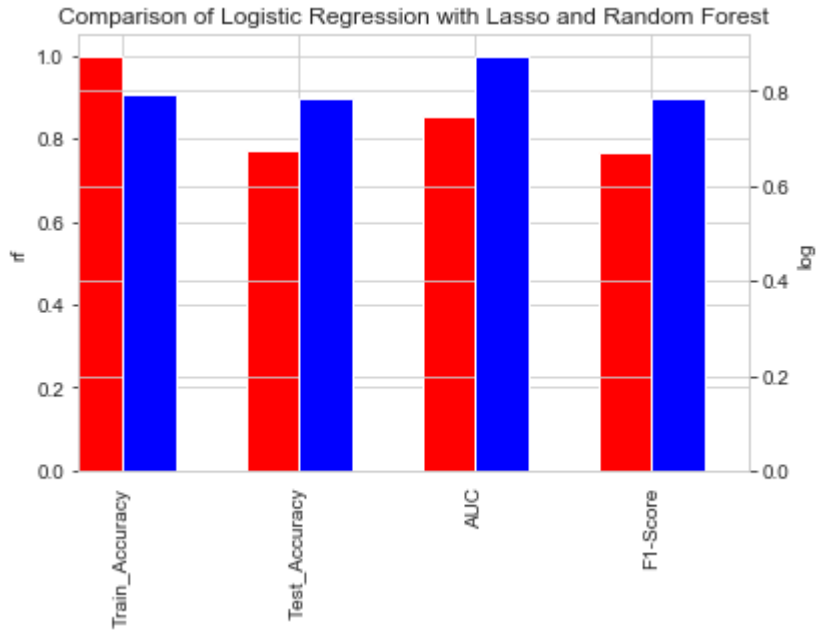
In [40]:
```
rf_train_score = rf.score(X_train,y_train)
rf_test_score = rf.score(X_test,y_test)
rf_auc = metrics.roc_auc_score(y_test, prob_retention_rf)
rf_metrics = pd.DataFrame([rf_train_score, rf_test_score, rf_auc, prfs[2]])

log_train_score = logmodel_cv_l1.score(X_train,y_train)
log_test_score = logmodel_cv_l1.score(X_test,y_test)
log_auc = metrics.roc_auc_score(y_test, prob_retention_cv_l1)
log_metrics = pd.DataFrame([log_tr ain_score, log_test_score, log_auc, log_prf
s[2]])
```

In [41]:
```
metric=pd.DataFrame(index= np.arange(0,len(rf_metrics),1))
# rf_metrics.columns = ['rf']
# log_metrics.columns = ['log']
metric['rf']=rf_metrics
metric['log']=log_metrics
metric.index= ['Train_Accuracy', 'Test_Accuracy', 'AUC', 'F1-Score']
metric
```

Out[41]:

| | rf | log |
|---|---|---|
| **Train_Accuracy** | 0.999335 | 0.792568 |
| **Test_Accuracy** | 0.770731 | 0.786801 |
| **AUC** | 0.855452 | 0.874344 |
| **F1-Score** | 0.767236 | 0.784686 |

In [42]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax2 = ax.twinx()
width = 0.3
metric.rf.plot(kind='bar', color='red', ax=ax, width=width, position=1)
metric.log.plot(kind='bar', color='blue', ax=ax2, width=width, position=0)
ax.set_ylabel('rf')
ax2.set_ylabel('log')
plt.title('Comparison of Logistic Regression with Lasso and Random Forest')
plt.show()
```



# Final Model Selection:

As a results of the bar chart, it is clear that more sofisticated model of random forest with larger number of trees is not required. Final selected model is Logistic regression with lasso regularization for predicting whether a person would quirt and the Ridge model for finding the main drives of retention. It shows the importance of features as folowing bar chart. The advantage of logistic regression with regularization is that we can rely on the direction of parameters unlike in the unreqularized logistic regression model.

For instance:

- those who have been working for a longer time are more likely to quit, where as those who have higher salary are less likely to quit.
- it is important to note that Random forest although provides important drivers of retention but do not indicate the direction.
- if an employee is in Sales, Marketing, and Custome Service deprtments they are more likely to quit compared to other departments, if they are from engineering and datascience department they are less likely to quit.
- there is evidence inthe data that company id 1 and 2 have employees are also kess likely to quit.

The results do make sence and predict 78% of total number of those who have quit in the test data and also 78% of those who have been predicted to quit are actually those who quit (recall and precision).

In [43]:
```
logistic_impo_vars = []
color=[]
for i in range(len(parameters_cv_l2[0])):
    if parameters_cv_l2[0][i] != 0:
        logistic_impo_vars.append(X_train.columns[i])
        print(parameters_cv_l2[0][i])
    if parameters_cv_l2[0][i] < 0:
        color.append((0.9,0.1,0.3,0.8))
    else:
        color.append((0.1,0.5,0.9,0.8))
```

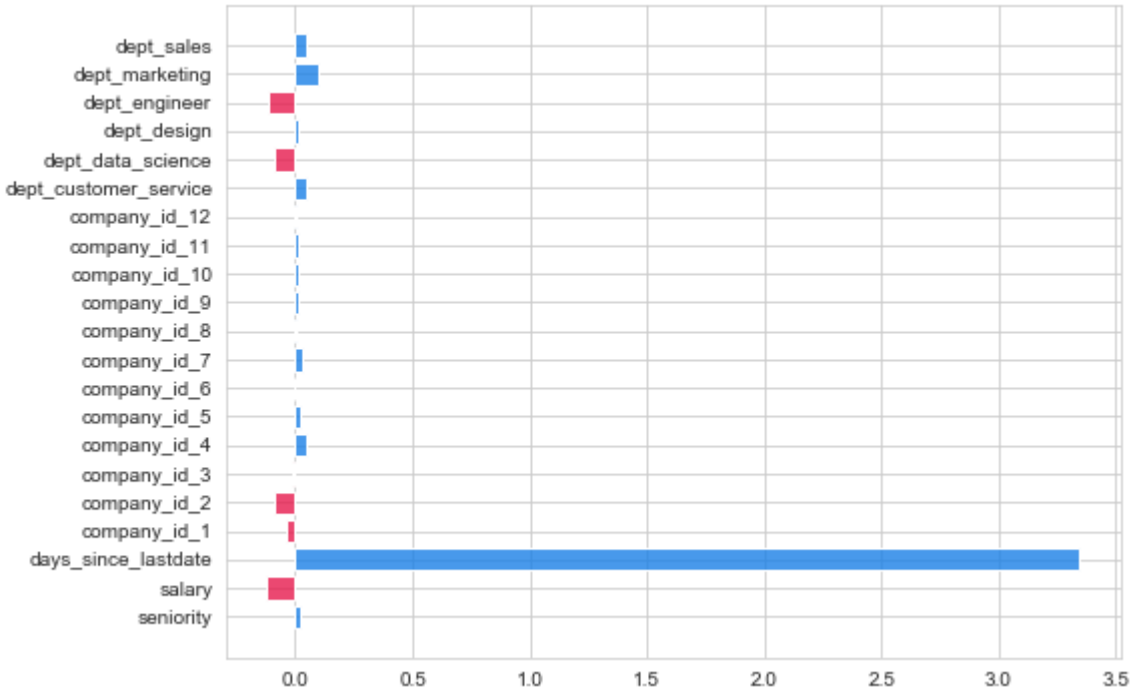
```
0.0222557951410623
-0.11960633538962867
3.3464508842454808
-0.03588481626317798
-0.08770220964852851
-0.01305264418009237
0.04613439160763152
0.02459441948226879
-0.005070345746470651
0.03166372424810043
0.002782579496281701
0.010510983930659181
0.01468966211164288
0.010455363135767426
0.001196703445149902
0.04331696803900671
-0.08632797938145638
0.010381545043623843
-0.11778899583587804
0.10235861643184263
0.04837765732209368
```

In [44]:
```
logistic_impo_vars
```

Out[44]:
```
['seniority',
 'salary',
 'days_since_lastdate',
 'company_id_1',
 'company_id_2',
 'company_id_3',
 'company_id_4',
 'company_id_5',
 'company_id_6',
 'company_id_7',
 'company_id_8',
 'company_id_9',
 'company_id_10',
 'company_id_11',
 'company_id_12',
 'dept_customer_service',
 'dept_data_science',
 'dept_design',
 'dept_engineer',
 'dept_marketing',
 'dept_sales']
```

In [45]:
```
plt.figure(figsize = (8,6))
plt.barh(X_train.columns, parameters_cv_l2[0],color=color, )
```

Out[45]: <BarContainer object of 21 artists>




# Answer to Questions:

- What might you be able to do for the company to address employee Churn, what would be follow-up actions?

An investigattion in the sales, marketing, customer service departments. An annonimous survay of the issues or challenges that employees may have at these departments may reveal some of the resons for what employees in thse departments are more likely to quit.

- If you could add to this data set just one variable that could help explain employee churn, what would that be?

I would ask for data on number of times each employee has been promoted. This data may be correlatedwith days-since_lastdate but using regularized models we can find the exact amount of coefficients associated to each variable. it can also carry more information than days-since_lastdate, therefore the accuracy and AUC is likely to increase.