DataChallenge1

February 20, 2019

GOAL Employee turnover is a very costly problem for companies. The cost of replacing an employee if often larger than 100K USD, taking into account the time spent to interview and find a replacement, placement fees, sign-on bonuses and the loss of productivity for several months. t is only natural then that data science has started being applied to this area. Understanding why and when employees are most likely to leave can lead to actions to improve employee retention as well as planning new hiring in advance. This application of DS is sometimes called people analytics or people data science (if you see a job title: people data scientist, this is your job). In this challenge, you have a data set with info about the employees and have to predict when employees are going to quit by understanding the main drivers of employee churn.

DATA We got employee data from a few companies. We have data about all employees who joined from 2011/01/24 to 2015/12/13. For each employee, we also know if they are still at the company as of 2015/12/13 or they have quit. Beside that, we have general info about the employee, such as avg salary during her tenure, dept, and yrs of experience. As said above, the goal is to predict employee retention and understand its main drivers.

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

In [2]:
```
# load data
df = pd.read_csv('employee_retention_data.csv')
```

In [3]:
```
# explore data
print('shape: ',df.shape)
df.head()
```

```
shape:  (24702, 7)
```

Out[3]:
```
   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
```

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

In [ ]:
```
# seems like we could engineer a feature to see time at company.
```

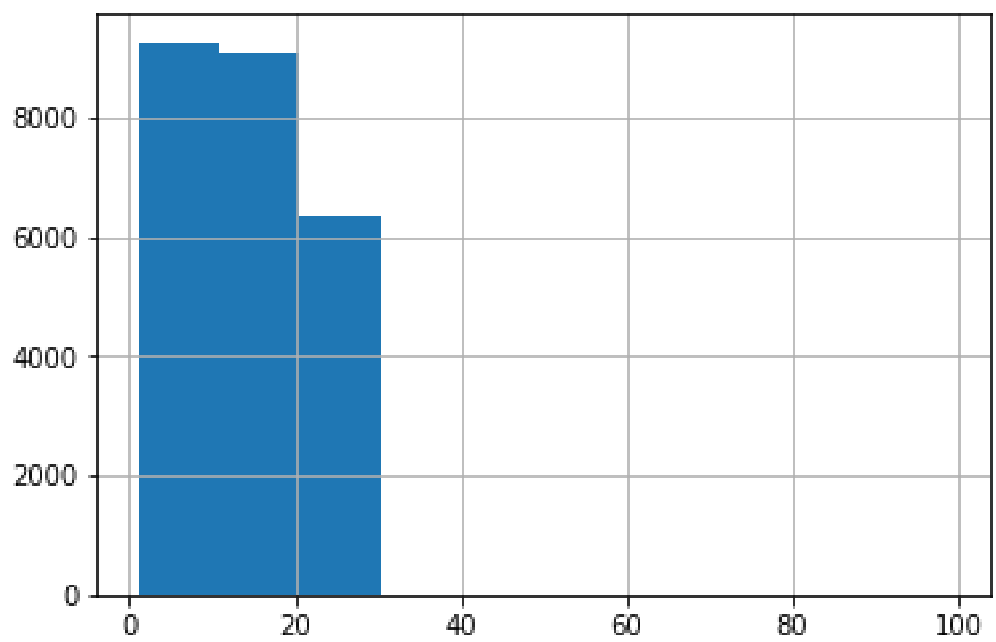
In [4]:
```
# define quitters
df.loc[:, 'quit'] = 1
df.loc[df['quit_date'].isnull(), 'quit'] = 0
```

In [5]:
```
# convert dates columns to datetime
df['join_date'] = pd.to_datetime(df['join_date'])
df['quit_date'] = pd.to_datetime(df['quit_date'])
```

In [6]:
```
# engineer 'time at co' feature - calc timedelta and convert to float
def f(row):
    """ given a row, function f will:
    calculate the total time spent at company (years) if employee has quit OR
    calculate time spent at company (years) so far, based on date given in preface"""
    if row['quit'] == 1:
        val = ((row['quit_date'] - row['join_date'])/ np.timedelta64(1, 'Y'))
    elif row['quit'] == 0:
        val = ((pd.to_datetime('2015-12-13') - row['join_date'])/ np.timedelta64(1, 'Y
    return val

df['time_at_Co'] = df.apply(f, axis=1)
```

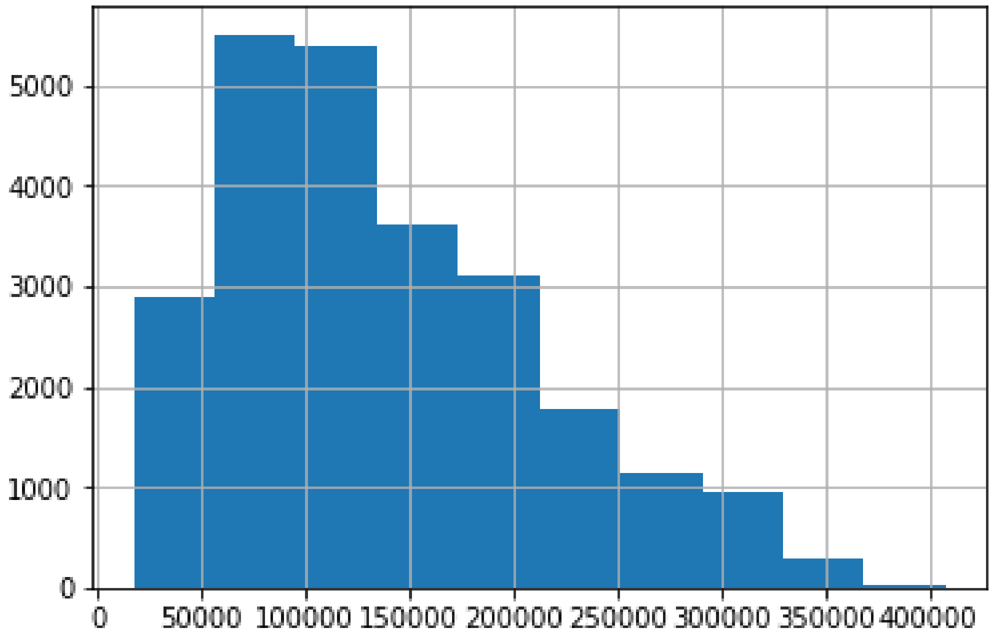
In [7]:
```
df.seniority.hist()
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb92e5ceda0>
```



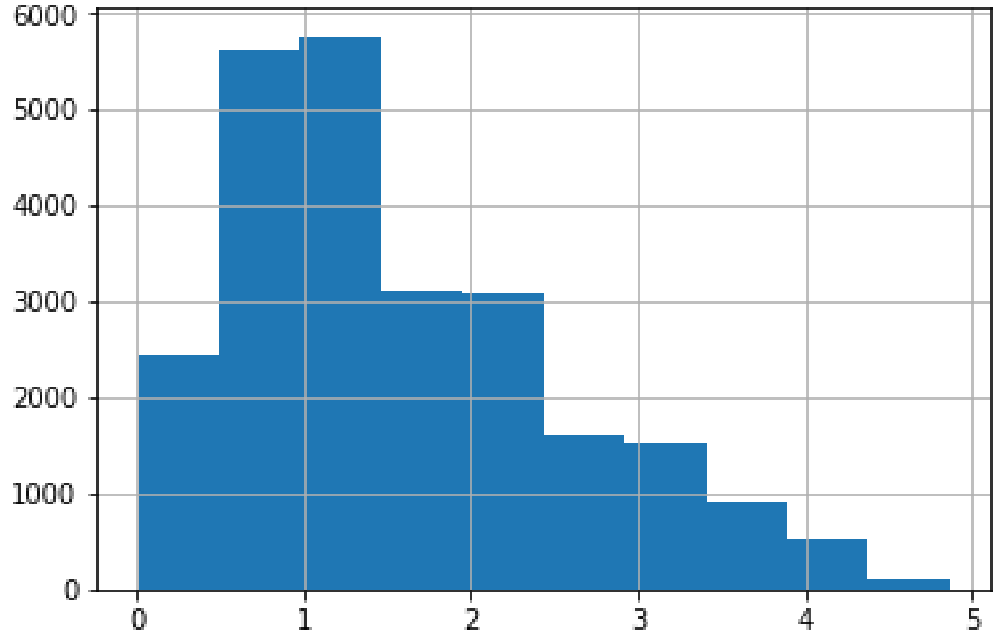
```
In [8]: df.salary.hist()
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e30dcf8>
```




```
In [9]: df.time_at_Co.hist()
```

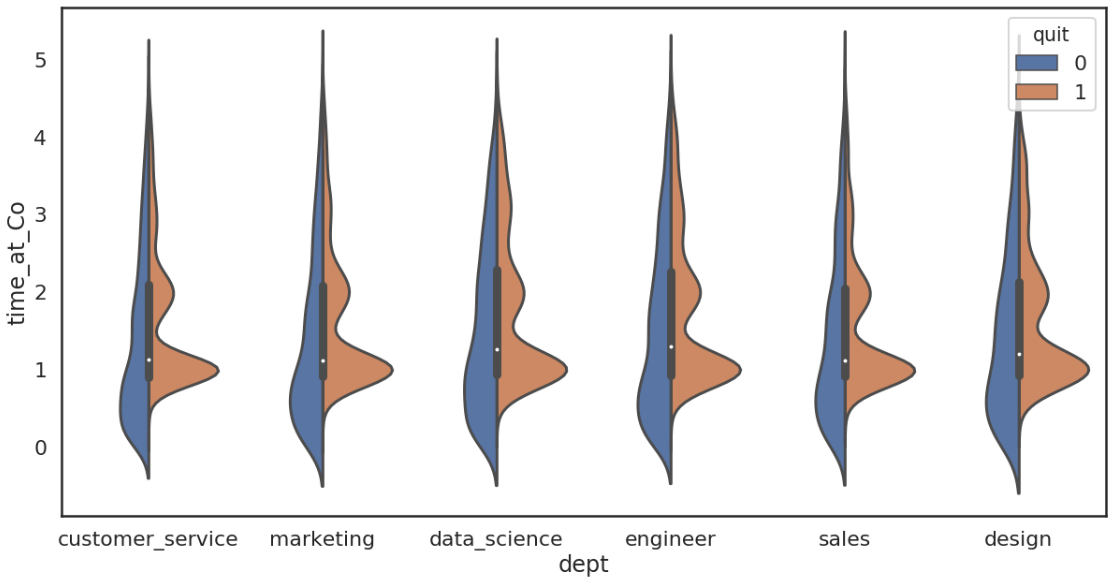
```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e324160>
```




```
In [10]: # let's look at time_at_Co by dept, split by quit status
         sns.set(context = 'poster', style = 'white')
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'dept', y = 'time_at_Co',  hue = 'quit', data = df, split=True)
```

```
/home/jim/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

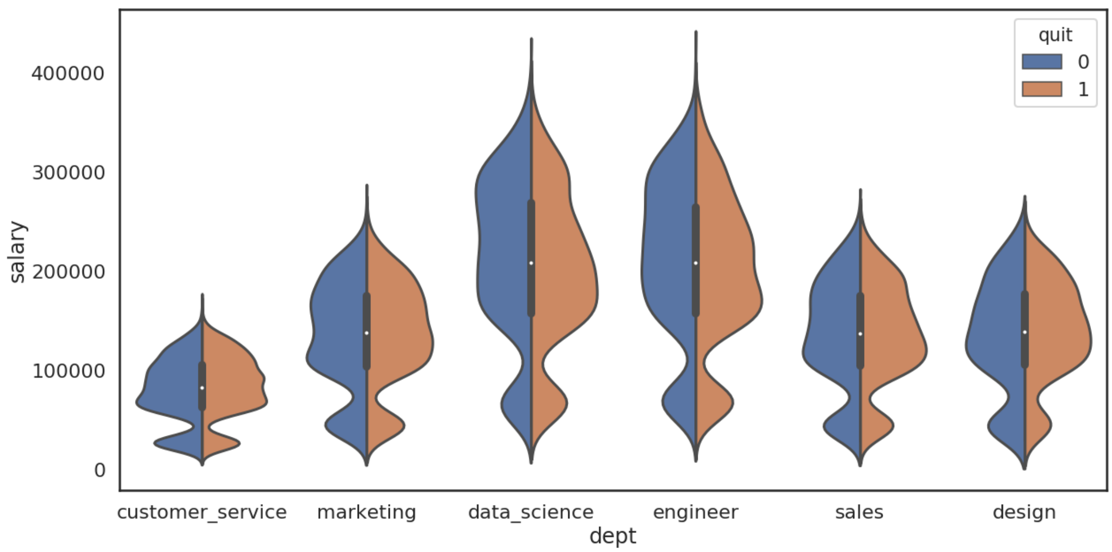
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e250fd0>
```



```
In [ ]: # looks like a lot of people leave at 1 year and again at 2 years.
        # not a lot of people quit before 1 year.
```

```
In [11]: # look at salaries by dept, split by quit
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'dept', y = 'salary',  hue = 'quit', data = df, split=True)
```

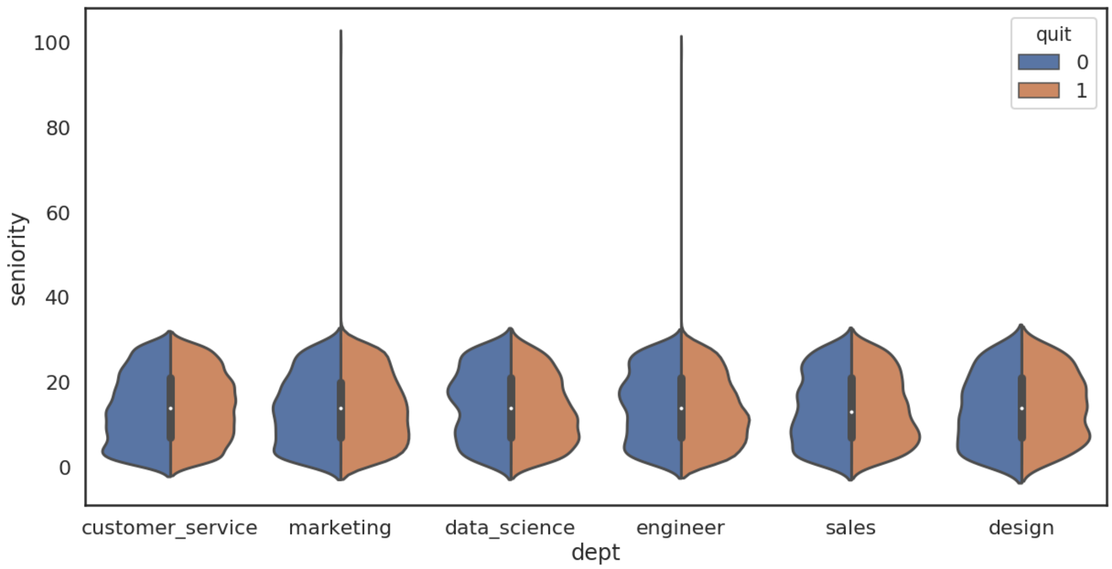
```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e21ba90>
```



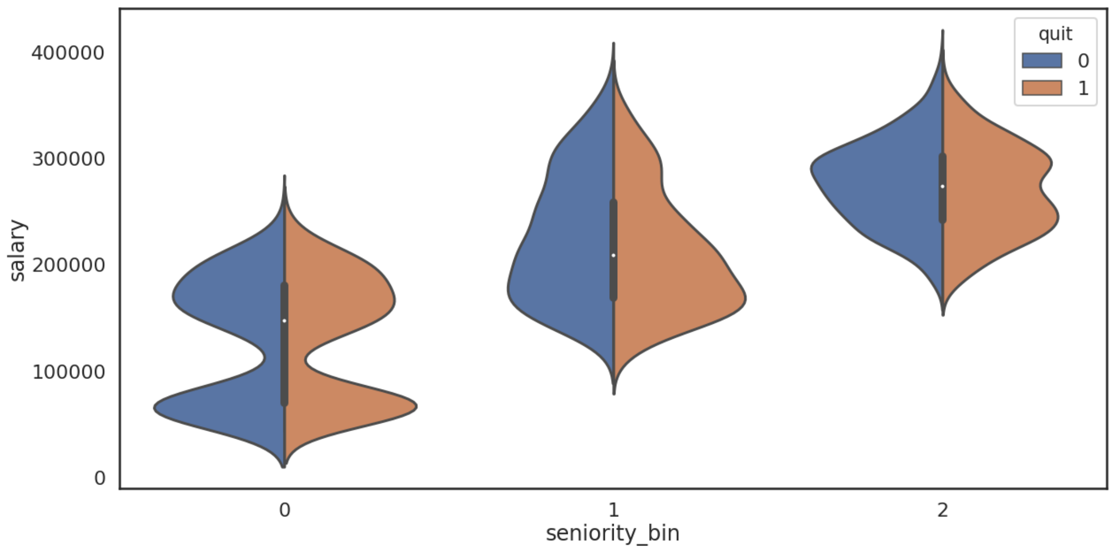

```
In [ ]: # might play some part, but not as stark as time_at_Co
```

```
In [12]: # look at experience by dept, split by quit
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'dept', y = 'seniority',  hue = 'quit', data = df, split=True)
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e30dc18>
```



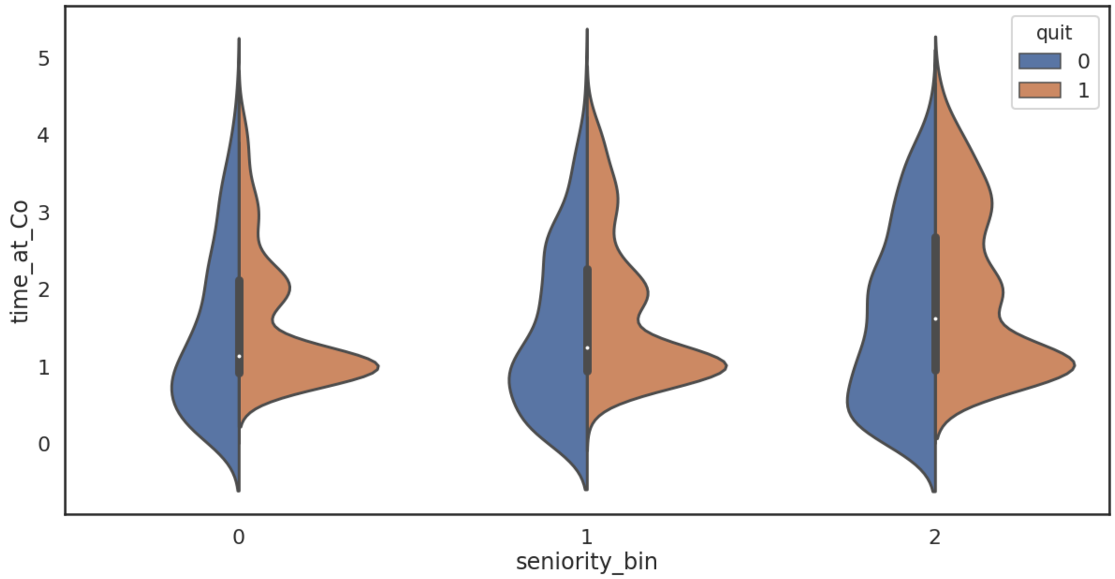

```
In [ ]: # looks like some outliers in marketing and engineering
        # might need to tweak these features to bins instead of continuous

In [13]: # bin seniority into 3 bins = idea being junior, mid, and senior
         df['seniority_bin'] = pd.qcut(df.seniority, 3, labels=False)

In [14]: # look at salary per experience amoung data scientists, split by quit
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'seniority_bin', y = 'salary',  hue = 'quit', data = df.loc[df['de

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e0d1278>
```



```
In [ ]: # looks to some slight differences in salary between more senior quitters, but not at 
```

```
In [15]: # look at time_at_Co per experience amoung data scientists
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'seniority_bin', y = 'time_at_Co',  hue = 'quit', data = df.loc[df
```

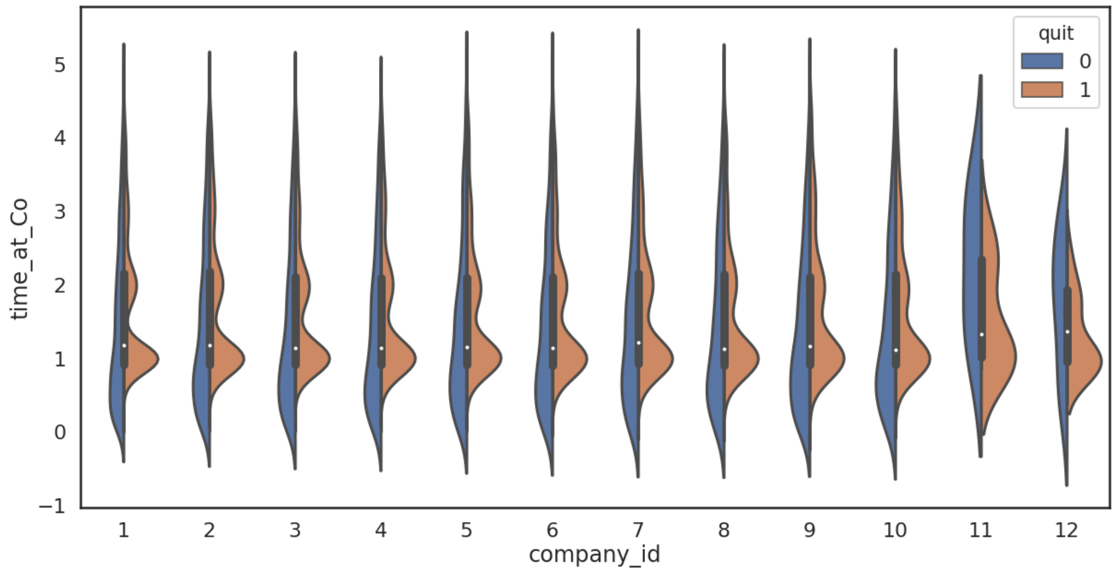
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92e05aef0>
```




```
In [ ]: # regardless of experience, bulk of quitters leave at 1 year, with spikes at 2 and 3 y
        # my intutition is that this is enough time for individuals to test the company for fi
        # or maybe they found a better opportunity.
```

```
In [16]: # something else I thought - maybe company matters?
         # look at salary_diff per company_id
         plt.figure(figsize=(20,10))
         sns.violinplot(x = 'company_id', y = 'time_at_Co',  hue = 'quit', data = df , split=T
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb92defc240>
```




```
In [17]: # drop employee id, not important for prediction
         df.drop(['employee_id'], axis=1, inplace=True)

In [18]: df.head()

Out[18]:    company_id              dept  seniority    salary  join_date  quit_date  \
         0           7  customer_service         28   89000.0 2014-03-24 2015-10-30
         1           7         marketing         20  183000.0 2013-04-29 2014-04-04
         2           4         marketing         14  101000.0 2014-10-13        NaT
         3           7  customer_service         20  115000.0 2012-05-14 2013-06-07
         4           2      data_science         23  276000.0 2011-10-17 2014-08-22

            quit  time_at_Co  seniority_bin
         0     1    1.601676              2
         1     1    0.930888              2
         2     0    1.166348              1
         3     1    1.065046              2
         4     1    2.847423              2

In [19]: # label encode dept
         from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         df['dept'] = le.fit_transform(df['dept'])

In [20]: df.head()

Out[20]:    company_id  dept  seniority    salary  join_date  quit_date  quit  \
         0           7     0         28   89000.0 2014-03-24 2015-10-30     1
         1           7     4         20  183000.0 2013-04-29 2014-04-04     1
```

```
2           4     4         14  101000.0 2014-10-13        NaT     0
3           7     0         20  115000.0 2012-05-14 2013-06-07     1
4           2     1         23  276000.0 2011-10-17 2014-08-22     1

   time_at_Co  seniority_bin
0    1.601676              2
1    0.930888              2
2    1.166348              1
3    1.065046              2
4    2.847423              2
```

In [21]:
```
# make sure there's no missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24702 entries, 0 to 24701
Data columns (total 9 columns):
company_id       24702 non-null int64
dept             24702 non-null int64
seniority        24702 non-null int64
salary           24702 non-null float64
join_date        24702 non-null datetime64[ns]
quit_date        13510 non-null datetime64[ns]
quit             24702 non-null int64
time_at_Co       24702 non-null float64
seniority_bin    24702 non-null int64
dtypes: datetime64[ns](2), float64(2), int64(5)
memory usage: 1.7 MB
```

In [22]:
```
# since I have the time_at_Co, and I'm not doing time series model, I think I can drop
# I'm also going to drop seniority, since I made the bins.
df.drop(['join_date', 'quit_date', 'seniority'], axis=1, inplace=True)
```

In [23]:
```
# since there's categorical data, I think a tree-based method would work well
```

In [34]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

# split into training and test data
y = df['quit'].values
X = df.drop(['quit'],axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=

rf = RandomForestClassifier(random_state = 42)

param_grid = {"n_estimators": [10, 20, 50, 100],
```

```
                  "max_depth": [2, 3, 4, None],
                  "max_features": [1, 2, 3, 'auto'],
                  "min_samples_split": [2, 5, 10, 20],
                  "bootstrap": [True, False],
                  "criterion": ["gini", "entropy"]}

    # run grid search
    grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5)
    grid_search.fit(X_train, y_train)
    grid_search.best_params_
```

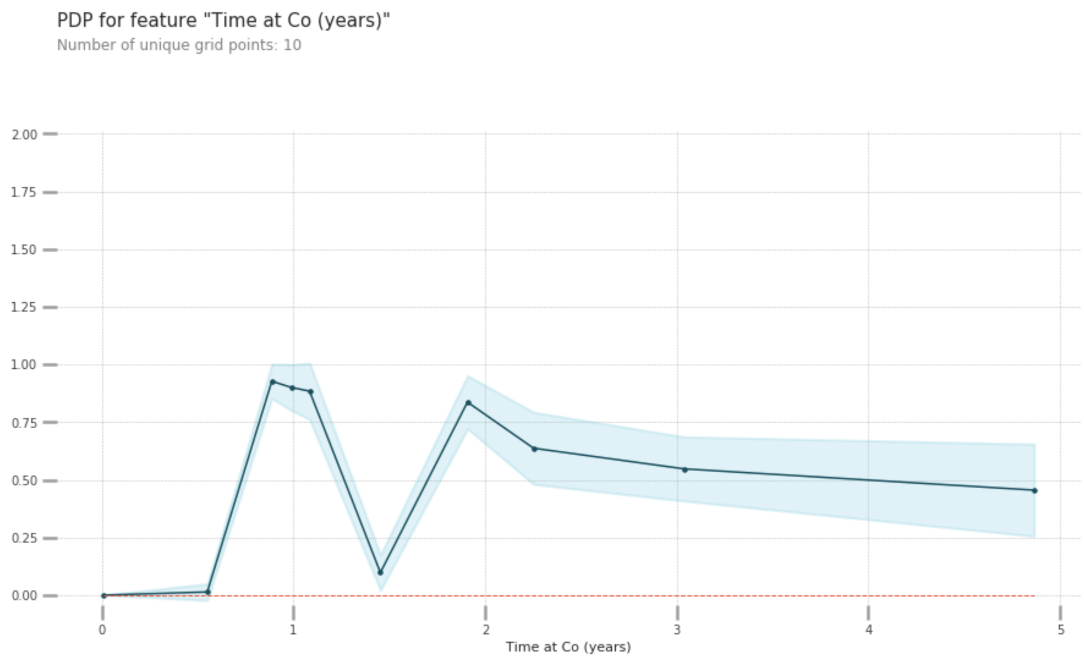
```
Out[34]: {'bootstrap': False,
          'criterion': 'entropy',
          'max_depth': None,
          'max_features': 3,
          'min_samples_split': 20,
          'n_estimators': 100}
```

```
In [38]: from sklearn import metrics

         # fit best model
         rf = RandomForestClassifier(n_estimators = 100,
                                     max_depth = None,
                                     max_features = 3,
                                     min_samples_split = 20,
                                     bootstrap = False,
                                     criterion = 'entropy',
                                     random_state = 42)
         rf.fit(X_train, y_train)
         y_pred = rf.predict(X_test)
         # Print the classification report
         target_names = ['quit', 'stay']
         print(metrics.classification_report(y_test, y_pred, target_names=target_names))
         print('AUC:',metrics.roc_auc_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        quit       0.83      0.75      0.79      4476
        stay       0.81      0.87      0.84      5405

   micro avg       0.82      0.82      0.82      9881
   macro avg       0.82      0.81      0.82      9881
weighted avg       0.82      0.82      0.82      9881

AUC: 0.8137132855339485
```

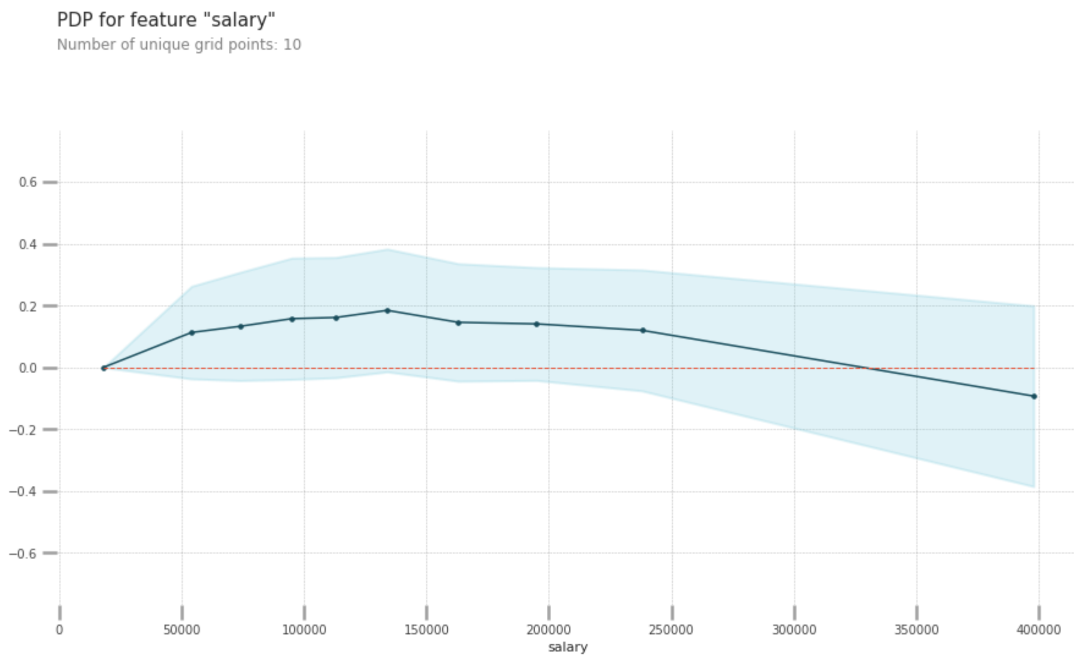
```
In [ ]: # .81 is good enough for now.
        # to explain feature importance, I'm going to use permutation, partial dependence plot
```

```
In [39]: # optional
         #!pip install eli5
         # get feature importances using permutation
         import eli5
         from eli5.sklearn import PermutationImportance

         perm = PermutationImportance(rf, random_state=42).fit(X_test, y_test)
         eli5.show_weights(perm, feature_names = X_test.columns.tolist())

Out[39]: <IPython.core.display.HTML object>

In [ ]: # confirms EDA, time at Co is super important for model predictions, followed by salar

In [41]: # optional install if needed
         #!pip install pdpbox
         from pdpbox import pdp, info_plots

         # see impact of salary
         pdp_ = pdp.pdp_isolate(model=rf, dataset=X_test, model_features=X_test.columns.tolist
         pdp.pdp_plot(pdp_, 'Time at Co (years)')
         plt.show()
```



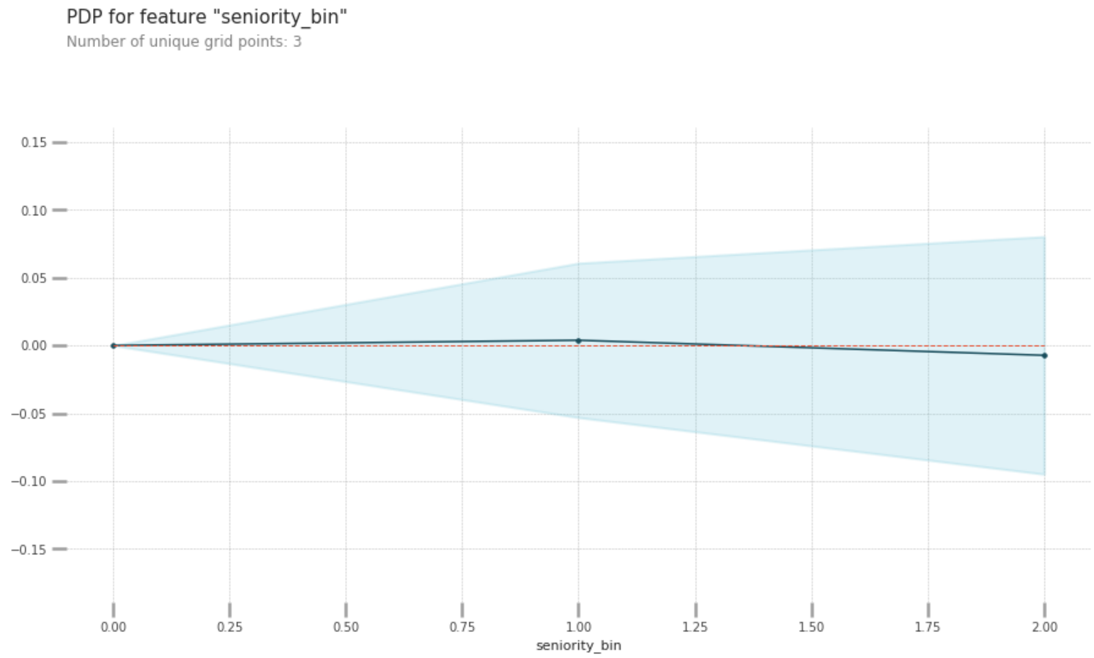

```
In [ ]: # interpretation: people start quitting once they've been at a job for <1 year.
        # 1 and 2 years heavily impact the prediction with little uncertainty.
```

In [42]:
```
# see impact of salary
pdp_ = pdp.pdp_isolate(model=rf, dataset=X_test, model_features=X_test.columns.tolist

# plot it
pdp.pdp_plot(pdp_, 'salary')
plt.show()
```



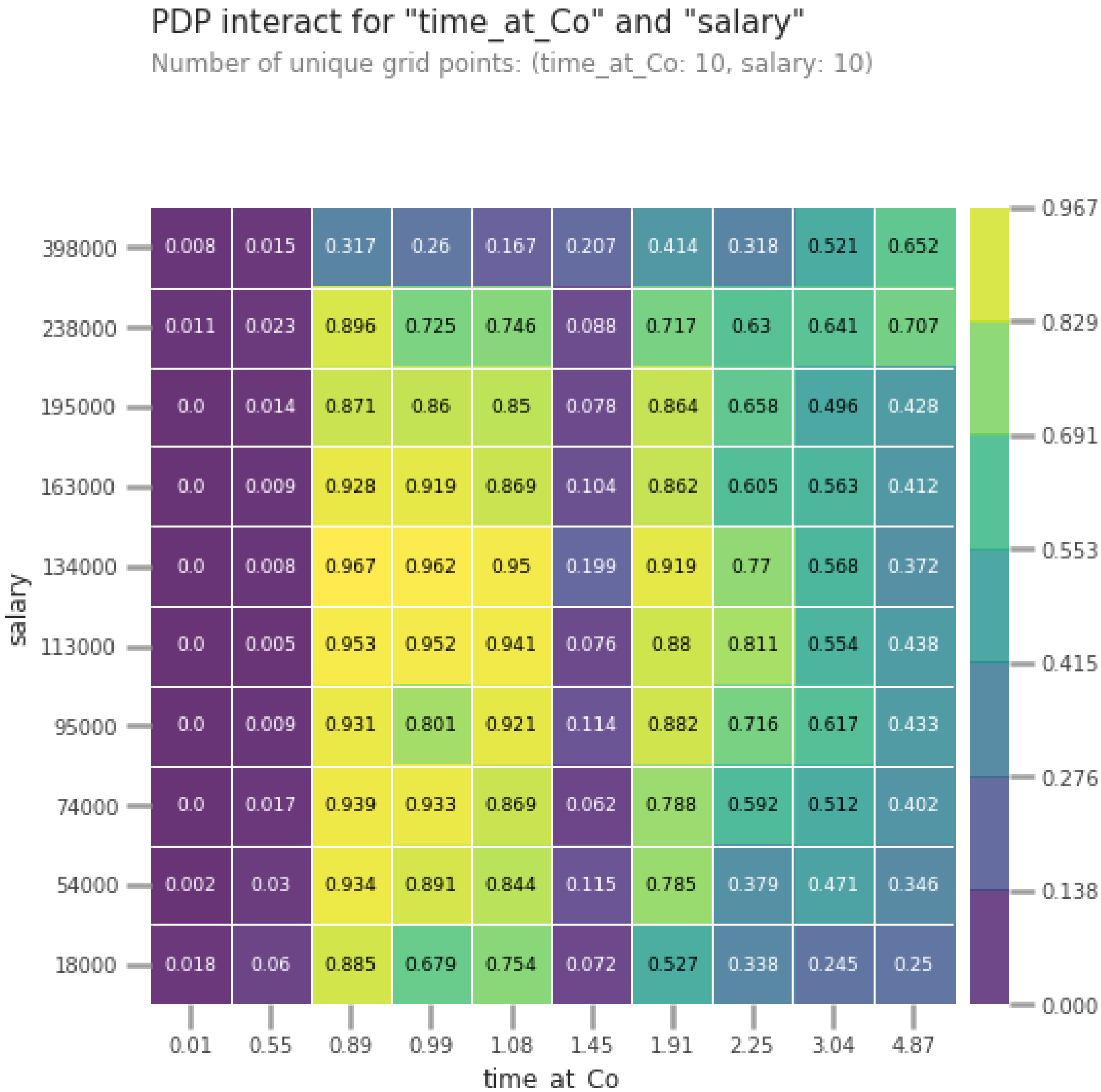
In [ ]:
```
# interpretation: there's a lot of uncertainty here. Salaries above $300000 negatively
```

In [43]:
```
# see impact of seniority
pdp_ = pdp.pdp_isolate(model=rf, dataset=X_test, model_features=X_test.columns.tolist

# plot it
pdp.pdp_plot(pdp_, 'seniority_bin')
plt.show()
```




```
In [ ]: # interpretation: seniority was not influential
```

```
In [46]: # impact of time vs salary
         features_to_plot = ['time_at_Co', 'salary']
         inter_ = pdp.pdp_interact(model=rf, dataset=X_test, model_features=X_test.columns.toli
         # grid
         pdp.pdp_interact_plot(pdp_interact_out=inter_, feature_names=features_to_plot, plot_ty
         plt.show()
```



In [ ]: # interpretation: time at the co dominates, although it you have a high salary it kind
        # regardless of salary,

In [ ]: # interpretation:
        # short time at company negatively impacts quitting prediction, but medium values posi
        # spending a long time at the company doesn't much matter
        # high salary scores negatively predict quitting
        # seniority doesn't have much impact

In [ ]: # OVERALL, time at the company appears to have strongest predictive value with respect
        # Given the unknowns regarding employee fit, role, satisifaction, opportunities...

```
# I would love to get my hands on exit interview responses.
# Answering WHY they are quitting may enable intervention.
```