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 em- ployee, 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

1

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>

2

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>

3

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 a non-tu

return np.add.reduce(sorted[indexer] \* weights, axis=axis) / sumval

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

4

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>

5

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['dept'] == 'd

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

6

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

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['dept'] =

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 years*

*# my intutition is that this is enough time for individuals to test the company for fit, role, t # 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=**True**)

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

7

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

8

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 join and*

*# 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=42)

rf = RandomForestClassifier(random\_state = 42)

param\_grid = {"n\_estimators": [10, 20, 50, 100],

9

"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 plots, and sha*

10

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 salary*

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(), featur 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.*

11

In [42]: *# see impact of salary*

pdp\_ = pdp.pdp\_isolate(model=rf, dataset=X\_test, model\_features=X\_test.columns.tolist(), featur

*# plot it* pdp.pdp\_plot(pdp\_, 'salary') plt.show()

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

In [43]: *# see impact of seniority*

pdp\_ = pdp.pdp\_isolate(model=rf, dataset=X\_test, model\_features=X\_test.columns.tolist(), featur

*# plot it* pdp.pdp\_plot(pdp\_, 'seniority\_bin') plt.show()

12

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.tolist(), fea *# grid* pdp.pdp\_interact\_plot(pdp\_interact\_out=inter\_, feature\_names=features\_to\_plot, plot\_type='grid' plt.show()

13

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

*# regardless of salary,*

In [ ]: *# interpretation:*

*# short time at company negatively impacts quitting prediction, but medium values positively inf # 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 respecting to chu*

*# Given the unknowns regarding employee fit, role, satisifaction, opportunities...*

14

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

15