data_generator

September 16, 2018

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

```
In [2]: all_columns = pd.read_excel("all_final_features.xlsx")
        all_columns.columns
```

```
Out[2]: Index(['ID', 'Gender', 'MaritalStatus', 'Education', 'CommuteDistance',
               'PerformanceRating', 'PercentSalaryHike', 'MonthlyIncome',
               'StockOptionLevel', 'CompanyProfit', 'Department', 'BusinessTravel',
               'YearsWithCurrentManager', 'YearsSinceLastPromotion', 'JobSatisfaction',
               'EnvironmentSatisfaction', 'TrainingHoursLastYear', 'TotalWorkingYears',
               'YearsInCurrentRole', 'YearsAtCompany', 'NumbersCompaniesWorked',
               'WorkLifeBal', 'JobLevel', 'Attrition', 'ReasonToManager',
               'ReasonInExitInterview'],
              dtype='object')
```

0.1 Data Patterns

0.1.1 Millennial

```
In [86]: ## Field Bounds ##
         num_samples = 7000

         age = "Millennial"

         attrition_val = ["Yes", "No"]
         attrition_rate = 0.45


         #Age Dependent

         ##interrelated
```

```
job_title = ["Associate Developer", "Developer"]
monthly_income =[ [40000, 50000], [45000, 60000]]
experience_company = [np.arange(0,4), np.arange(0, 7)]
curr_role = [np.arange(0,4), np.arange(0,4)] # less than experience_company
promotion_time = [np.arange(0,2), np.arange(0,4)] # less than experience_company
curr_manager =  [np.arange(0,4),np.arange(0,4)] # less than experience_company
stock_levels = [np.arange(0,2), np.arange(0,2)]
salary_hike = [np.arange(12,21), np.arange(10,19)]
experience = [np.arange(0,4), np.arange(0, 9)] #more than experience_company
num_companies = [np.arange(1,3), np.arange(1,5)]
travel = [np.arange(1), np.arange(0,2)]

## common in same age
form_values = np.arange(1,6)
form_values_prob = [0.3,0.1,0.2,0.1,0.3]
form_fields = ['JobSatisfaction','EnvironmentSatisfaction','WorkLifeBal']
training_times = np.arange(2,6)
training_times_prob = [0.15,0.25,0.2,0.4]
gender_prob = [0.5,0.5]
marital_status = ['Married', 'Single', 'Divorced']
marital_status_prob = [0.3,0.7,0]
education_prob = [0.1,0.9,0,0]
per_prob = [0,0,0.45,0.25,0.3]
commute_distance_prob = [0.6,0.3,0.1,0]

#Common Parameters
dept = ['Development', 'Testing', 'DevOps', 'Product Management', 'People Management'
        ,'Consultancy', 'Training']
company_profit = np.arange(-2,2)
gender = ['Male', 'Female']
education = ["Under-Graduate","Graduate","Post-Graduate", "Doctorate"]
commute_distance = np.arange(1,5)
perf_rating = np.arange(1,6)

# Attrition yes
patterns = pd.read_excel("all_final_features.xlsx", sheet_name="millennial_patterns")
reasons = patterns[["ReasonToManager", "ReasonInExitInterview"]]
reason_manager = np.array(reasons["ReasonToManager"])
reason_manager = reason_manager.tolist()
reason_exit = np.array(reasons["ReasonInExitInterview"])
reason_exit = reason_exit.tolist()
reasons_prob = [0.4,0.3,0.3]
```

0.1.2 GenX

In [37]:
```
## Field Bounds ##
num_samples = 7000
```

```
age = "Genx"

attrition_val = ["Yes", "No"]
attrition_rate = 0.4


#Age Dependent

##interrelated
job_title = ["Developer", "Consultant", "Senior Developer", "Senior Consultant"]
monthly_income =[ [45000, 60000], [45000, 60000], [55000, 85000], [55000, 85000]]
experience_company = [np.arange(0,8),np.arange(0,8), np.arange(0, 10), np.arange(0, 1
curr_role = [np.arange(3,8), np.arange(3,8), np.arange(2,5), np.arange(2,5)] # less t
promotion_time = [np.arange(0,3), np.arange(0,3), np.arange(0,6), np.arange(0,6)] # l
curr_manager =  [np.arange(0,4), np.arange(0,4), np.arange(0,6), np.arange(0,6)] # le
stock_levels = [np.arange(1,3), np.arange(1,3), np.arange(2,3), np.arange(2,3)]
salary_hike = [np.arange(12,19), np.arange(12,19), np.arange(10,15), np.arange(10,15)]
experience = [np.arange(5,10), np.arange(5,10), np.arange(8,13), np.arange(8,13)] #mo
num_companies = [np.arange(0,5), np.arange(0,5), np.arange(1,5), np.arange(1,5)]
travel = [np.arange(1,2), np.arange(1,2), np.arange(2,3), np.arange(2,3)]

## common in same age
form_values = np.arange(1,6)
form_values_prob = [0.2,0.1,0.3,0.3,0.1]
form_fields = ['JobSatisfaction','EnvironmentSatisfaction','WorkLifeBal']
training_times = np.arange(1,5)
training_times_prob = [0.25,0.3,0.3,0.15]
gender_prob = [0.6,0.4]
marital_status = ['Married', 'Single', 'Divorced']
marital_status_prob = [0.6,0.3,0.1]
education_prob = [0,0.6,0.4,0]
per_prob = [0,0,0.4,0.35,0.25]
commute_distance_prob = [0.3,0.35,0.15,0.2]

#Common Parameters
dept = ['Development', 'Testing', 'DevOps', 'Product Management', 'People Management'
        ,'Consultancy', 'Training']
company_profit = np.arange(-2,2) # Dependent on salary hike & attrition
gender = ['Male', 'Female']
education = ["Under-Graduate","Graduate","Post-Graduate", "Doctorate"]
commute_distance = np.arange(1,5)
perf_rating = np.arange(1,6)

# Attrition yes
patterns = pd.read_excel("all_final_features.xlsx", sheet_name="genx_patterns")
reasons = patterns[["ReasonToManager", "ReasonInExitInterview"]]
reasons_prob = [0.4,0.3,0.3]
```

```
In [38]: reasons
```

```
Out[38]:             ReasonToManager      ReasonInExitInterview
         0  Family Responsibilities     Rigid Maternity Policy
         1                   Growth  Better career opportunity
         2         Business Travel             Business Travel
```

0.1.3 BabyBoomers

```
In [5]: ## Field Bounds ##
        num_samples = 7000

        age = "BabyBoomers"

        attrition_val = ["Yes", "No"]
        attrition_rate = 0.38


        #Age Dependent

        ##interrelated
        job_title = ["Senior Developer", "Product Manager", "Product Owner", "VP"]
        monthly_income =[ [45000, 65000], [45000, 65000], [75000, 95000], [75000, 95000]]
        experience_company = [np.arange(5, 10), np.arange(8, 12), np.arange(9, 13), np.arange(
        curr_role = [np.arange(3,6), np.arange(3,8), np.arange(3,9), np.arange(3,12)] # less t
        promotion_time = [np.arange(4), np.arange(6), np.arange(6), np.arange(7)] # less than
        curr_manager =  [np.arange(4),np.arange(2,8),np.arange(2,8),np.arange(3,10)] # less th
        stock_levels = [np.arange(2,3), np.arange(2,3), np.arange(2,4), np.arange(2,4)]
        salary_hike = [np.arange(12,19), np.arange(8,13), np.arange(6,10), np.arange(6,10)]
        experience = [np.arange(8,21), np.arange(8,21), np.arange(8,21), np.arange(15,21)] #mo
        num_companies = [np.arange(1,5), np.arange(1,5), np.arange(2,5), np.arange(2,5)]
        travel = [np.arange(2,4), np.arange(2,4), np.arange(3,4), np.arange(3,4)]

        ## common in same age
        form_values = np.arange(1,6)
        form_values_prob = [0.15,0.2,0.2,0.25,0.2]
        form_fields = ['JobSatisfaction','EnvironmentSatisfaction','WorkLifeBal']
        training_times = np.arange(1,5)
        training_times_prob = [0.3,0.25,0.15,0.3]
        gender_prob = [0.6,0.4]
        marital_status = ['Married', 'Single', 'Divorced']
        marital_status_prob = [0.75,0.15,0.1]
        education_prob = [0,0.5,0.4,0.1]
        per_prob = [0,0,0.4,0.35,0.25]
        commute_distance_prob = [0.2,0.25,0.25,0.3]

        #Common Parameters
        dept = ['Development', 'Testing', 'DevOps', 'Product Management', 'People Management',
```

```
        ,'Consultancy', 'Training']
company_profit = np.arange(-2,2) # Dependent on salary hike & attrition
gender = ['Male', 'Female']
education = ["Under-Graduate","Graduate","Post-Graduate", "Doctorate"]
commute_distance = np.arange(1,5)
perf_rating = np.arange(1,6)
```

0.2 Generator Functions

```
In [87]: all_data = pd.read_excel("all_final_features.xlsx",sheet_name="all_data")
         all_data.columns
```

```
Out[87]: Index(['ID', 'Gender', 'MaritalStatus', 'Education', 'CommuteDistance',
                'PerformanceRating', 'PercentSalaryHike', 'MonthlyIncome',
                'StockOptionLevel', 'CompanyProfit', 'Department', 'BusinessTravel',
                'YearsWithCurrentManager', 'YearsSinceLastPromotion', 'JobSatisfaction',
                'EnvironmentSatisfaction', 'TrainingHoursLastYear', 'TotalWorkingYears',
                'YearsInCurrentRole', 'YearsAtCompany', 'NumbersCompaniesWorked',
                'WorkLifeBal', 'JobLevel', 'Attrition', 'ReasonToManager',
                'ReasonInExitInterview'],
               dtype='object')
```

```
In [88]: def assign_attrition(u):
             if(np.random.uniform() < attrition_rate):
                 return "No"
             return "Yes"
         attrition_col = pd.Series(data = np.zeros(num_samples))
         all_data["Attrition"] = attrition_col
         all_data["Attrition"] = all_data["Attrition"].apply(assign_attrition)
```

```
In [89]: attr_yes = all_data[all_data["Attrition"] == "Yes"]
         attr_no = all_data[all_data["Attrition"] == "No"]
```

```
In [90]: def assign_job_title(u):
             return np.random.choice(job_title)
         all_data["JobLevel"] = all_data["JobLevel"].apply(assign_job_title)
```

```
In [91]: def assign_interrelated_job(x,z):
             index = job_title.index(x)
             return np.random.choice(z[index])
```

```
In [92]: interrelated_job = [stock_levels, salary_hike, promotion_time, num_companies, travel]
         all_data["StockOptionLevel"] = all_data.apply(lambda row: assign_interrelated_job(row
         all_data["PercentSalaryHike"] = all_data.apply(lambda row: assign_interrelated_job(ro
         all_data["YearsSinceLastPromotion"] = all_data.apply(lambda row: assign_interrelated_
         all_data["NumbersCompaniesWorked"] = all_data.apply(lambda row: assign_interrelated_j
         all_data["BusinessTravel"] = all_data.apply(lambda row: assign_interrelated_job(row["
```

In [93]:
```
def assign_income(x):
    index = job_title.index(x)
    return np.random.choice(np.arange(monthly_income[index][0], monthly_income[index]
all_data["MonthlyIncome"] = all_data["JobLevel"].apply(assign_income)
```

In [94]:
```
def assign_experience(x):
    index = job_title.index(x)
    return np.random.choice(experience[index])
all_data["TotalWorkingYears"] = all_data["JobLevel"].apply(assign_experience)
```

In [95]:
```
def assign_company(x,y):
    index = job_title.index(x)
    if(min(experience_company[index]) == min(y,max(experience_company[index]))):
        return min(y,max(experience_company[index]))
    else:
        return np.random.choice(np.arange(min(experience_company[index]),min(y,max(ex

all_data["YearsAtCompany"] = all_data.apply(lambda row: assign_company(row["JobLevel"]
```

In [96]:
```
def assign_numcompanies(x,y,z):
    if x == y:
        return 1
    else:
        index = job_title.index(z)
        return np.random.choice(num_companies[index])
all_data["NumbersCompaniesWorked"] = all_data.apply(lambda row: assign_numcompanies(r
                                                                                    r
```

In [97]:
```
def assign_interrelated_yrscompany(x,y,z):
    index = job_title.index(x)
    u = np.random.choice(z[index])
    if u < y or u == y:
        return u
    elif min(z[index]) == min(y,max(z[index])):
        return min(z[index])
    else:
        return np.random.choice(np.arange(min(z[index]),min(y,max(z[index]))))
all_data["YearsWithCurrentManager"] = all_data.apply(lambda row: assign_interrelated_
                                                                    row["YearsAtComp
all_data["YearsInCurrentRole"] = all_data.apply(lambda row: assign_interrelated_yrscom
                                                                    row["YearsAtComp
all_data["YearsSinceLastPromotion"] = all_data.apply(lambda row: assign_interrelated_
                                                                    row["YearsAtComp
```

In [98]:
```
def prob_fields(values,prob):
    return np.random.choice(a=values,p=prob)
all_data['JobSatisfaction'] = all_data.apply(lambda row: prob_fields(form_values,form_
all_data['EnvironmentSatisfaction'] = all_data.apply(lambda row: prob_fields(form_valu
all_data['WorkLifeBal'] = all_data.apply(lambda row: prob_fields(form_values,form_valu
```

```
        all_data['TrainingHoursLastYear'] = all_data.apply(lambda row: prob_fields(training_t
        all_data['Gender'] = all_data.apply(lambda row: prob_fields(gender,gender_prob), axis=
        all_data['MaritalStatus'] = all_data.apply(lambda row: prob_fields(marital_status,mar
        all_data['Education'] = all_data.apply(lambda row: prob_fields(education,education_pr
        all_data['PerformanceRating'] = all_data.apply(lambda row: prob_fields(form_values,pe
        all_data['CommuteDistance'] = all_data.apply(lambda row: prob_fields(commute_distance
```

```
In [99]: def assign_companyprofit(x):
             return np.random.choice(company_profit)
         all_data["CompanyProfit"] = all_data["CompanyProfit"].apply(assign_companyprofit)
```

```
In [100]: def assign_dept(x):
              return np.random.choice(dept)
          all_data["Department"] = all_data["Department"].apply(assign_dept)
```

```
In [101]: attr_no = all_data.loc[all_data["Attrition"] == "Yes",:]

          def assign_reason_manager(x):
              return np.random.choice(a=reason_manager,p=reasons_prob)
          def assign_reason_exit(x):
              index = reason_manager.index(x)
              return reason_exit[index]
          attr_no["ReasonToManager"] = attr_no["ReasonToManager"].apply(assign_reason_manager)
          attr_no["ReasonInExitInterview"] = attr_no["ReasonToManager"].apply(assign_reason_ex

          all_data.loc[all_data["Attrition"] == "Yes",:] = attr_no
```

```
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:8: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:9: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [102]: #Reason 1: Repetitive task  Want Challenging task
          attr_perf_prob = [0,0,0,0.4,0.6]
          attr_job_prob = [0.1,0.3,0.5,0.1,0]
          # years in current role == years at company

          attr_no1 = attr_no[attr_no["ReasonToManager"] == reason_manager[0]]

          def assign_attr_perf(x):
```

```
            return np.random.choice(a=perf_rating,p=attr_perf_prob)
        attr_no1["PerformanceRating"] = attr_no1["PerformanceRating"].apply(assign_attr_perf)

        def assign_job_sat(x):
            return np.random.choice(a=form_values, p=attr_job_prob)
        attr_no1["JobSatisfaction"] = attr_no1["JobSatisfaction"].apply(assign_job_sat)

        attr_no[attr_no["ReasonToManager"] == reason_manager[0]] = attr_no1
```

```
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:10:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  # Remove the CWD from sys.path while we load stuff.
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:14:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:16:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  app.launch_new_instance()
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexing.py:54
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

```
In [103]: #Reason 2: Family Responsibilities Rigid Maternity Policy
          attr_gender = "Female"
          attr_marital_status = "Married"
          attr_job_prob = [0,0,0.2,0.5,0.3]
          attr_perf_prob = [0,0.1,0.4,0.4,0.1]

          attr_no2 = attr_no[attr_no["ReasonToManager"] == reason_manager[1]]

          def assign_attr_gender(x):
              return attr_gender

          def assign_attr_marital(x):
              return attr_marital_status
```

```
        def assign_attr_perf(x):
            return np.random.choice(a=perf_rating,p=attr_perf_prob)

        def assign_attr_job(x):
            return np.random.choice(a=form_values, p=attr_job_prob)

        attr_no2["Gender"] = attr_no2["Gender"].apply(assign_attr_gender)
        attr_no2["MaritalStatus"] = attr_no2["MaritalStatus"].apply(assign_attr_marital)
        attr_no2["PerformanceRating"] = attr_no2["PerformanceRating"].apply(assign_attr_perf)
        attr_no2["JobSatisfaction"] = attr_no2["JobSatisfaction"].apply(assign_attr_job)
        attr_no[attr_no["ReasonToManager"] == reason_manager[1]] = attr_no2
```

```
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:21:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:22:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:23:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:24:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:25:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexing.py:54
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

```
In [104]: #Reason 3: Technology Higher studies
          attr_training_prob = [0,0.1,0.4,0.5]
```

```
        attr_perf_prob = [0,0,0,0.3,0.7]
        attr_marital_status = "Single"

        attr_no3 = attr_no[attr_no["ReasonToManager"] == reason_manager[2]]

        def assign_attr_training(x):
            return np.random.choice(a=training_times,p=attr_training_prob)
        attr_no3["TrainingHoursLastYear"] = attr_no3["TrainingHoursLastYear"].apply(assign_at

        def assign_attr_perf(x):
            return np.random.choice(a=perf_rating,p=attr_perf_prob)
        attr_no3["PerformanceRating"] = attr_no3["PerformanceRating"].apply(assign_attr_perf)

        def assign_attr_marital(x):
            return attr_marital_status
        attr_no3["MaritalStatus"] = attr_no3["MaritalStatus"].apply(assign_attr_marital)
        attr_no[attr_no["ReasonToManager"] == reason_manager[2]] = attr_no3
```

```
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:10:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  # Remove the CWD from sys.path while we load stuff.
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:14:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:18:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:19:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
C:\Users\I349274\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexing.py:54
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

```
In [105]: all_data.loc[all_data["Attrition"] == "Yes",:] = attr_no
```

```
In [106]: all_data.to_excel("ttt.xlsx")
```