PREDICTING ABSENTEEISM

November 2019

Zachary Nguyen / Robert Mabry / Mohammad Qreyeah / Charlie Coffey

ABSENTEEISM

DEFINITION

Absence from work during normal working hours, resulting in temporary incapacity to execute regular working activity

THE PURPOSE

Explore whether a person presenting certain characteristics is expected to be away from work at some point in time or not

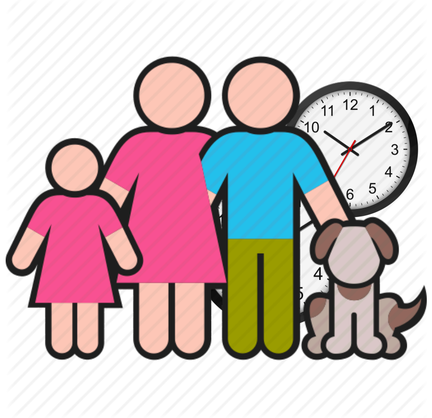
We want to know for how many working hours an employee could be away from work

ABSENTEEISM

Based on what information should we predict whether an employee is expected to be absent or not?

How far they live from the workplace

How many children and pets they have



Do they have higher education?

And so on!

THE DATASET

```
In [1]: import pandas as pd

In [2]: raw_csv_data = pd.read_csv("Absenteeism-data.csv")

In [3]: type(raw_csv_data)
Out[3]: pandas.core.frame.DataFrame

In [4]: raw_csv_data
Out[4]:
```

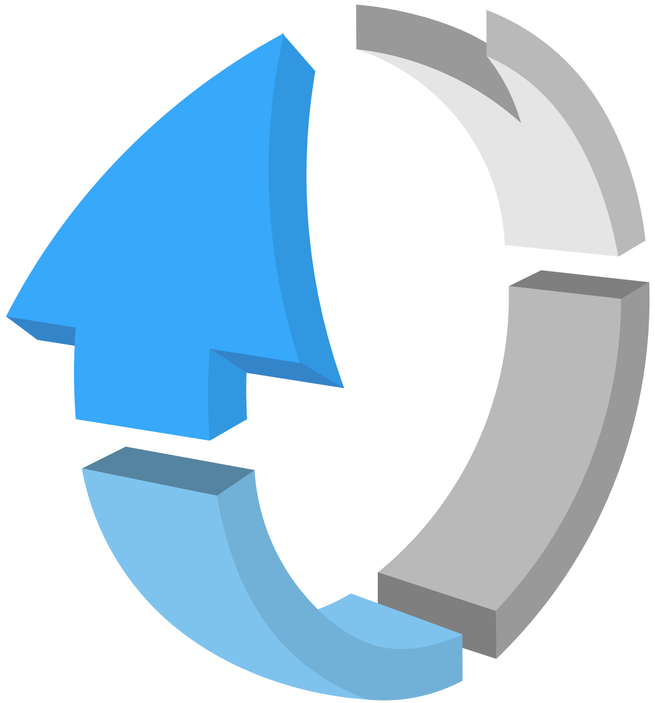
	ID	Reason for Absence	Date	Transportation Expense	Distance to Work	Age	Daily Work Load Average	Body Mass Index	Education	Children	Pets	Absenteeism Time in Hours
0	11	26	07/07/2015	289	36	33	239.554	30	1	2	1	4
1	36	0	14/07/2015	118	13	50	239.554	31	1	1	0	0
2	3	23	15/07/2015	179	51	38	239.554	31	1	0	0	2
3	7	7	16/07/2015	279	5	39	239.554	24	1	2	0	4
4	11	23	23/07/2015	289	36	33	239.554	30	1	2	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...
695	17	10	23/05/2018	179	22	40	237.656	22	2	2	0	8
696	28	6	23/05/2018	225	26	28	237.656	24	1	1	2	3
697	18	10	24/05/2018	330	16	28	237.656	25	2	0	0	8
698	25	23	24/05/2018	235	16	32	237.656	25	3	0	0	2
699	15	28	31/05/2018	291	31	40	237.656	25	1	1	1	2

700 rows × 12 columns

Dataset:

- Absenteeism_data.csv
- Already existing study about the prediction of absenteeism at work
- 100% application in the business world

WHAT TO EXPECT?



Data Preprocessing

Start working on the ‘Absenteeism_data.csv’ file and take it to a usable state in a machine learning algorithm.

Machine Learning

Develop a model that will predict the probability of an individual being excessively absent from work. For our case study, this will be a logistic regression model. Store the work as a Python module that we will call ‘absenteeism_module’ and will thus preserve it in a form suitable for further analysis.

Loading the ‘abesenteeism_module’

Load the ‘absenteeism_module’ and use its methods to obtain predictions.

Analyzing the predicted outputs in Tableau:

The visualizations we will obtain with this software will help us a great deal while looking for insights.

DATA PREPROCESCING

Cleaning the Reason for Absence

Reason for Absence	Date	Transportation Expense	Distance to Work	Age	Daily Work Ave
26	07/07/2015	289	36	33	23
0	14/07/2015	118	13	50	23
23	15/07/2015	179	51	38	23
7	16/07/2015	279	5	39	23
23	23/07/2015	289	36	33	23
23	10/07/2015	179	51	38	23
22	17/07/2015	361	52	28	23
23	24/07/2015	260	50	36	23
19	06/07/2015	155	12	34	23
22	13/07/2015	235	11	37	23

	Reason_1	Reason_2	Reason_3	Reason_4	Date	Tra
0	0	0	0	1	07/07/2015	
1	0	0	0	0	14/07/2015	
2	0	0	0	1	15/07/2015	
3	1	0	0	0	16/07/2015	
4	0	0	0	1	23/07/2015	

VARIOUS DISEASES	1	Certain infectious and parasitic diseases
	2	Neoplasms
	3	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
	4	Endocrine, nutritional and metabolic diseases
	5	Mental and behavioral disorders
	6	Diseases of the nervous system
	7	Diseases of the eye and adnexa
	8	Diseases of the ear and mastoid process
	9	Diseases of the circulatory system
	10	Diseases of the respiratory system
	11	Diseases of the digestive system
	12	Diseases of the skin and subcutaneous tissue
	13	Diseases of the musculoskeletal system and connective tissue
	14	Diseases of the genitourinary system
PREGNANCY	15	Pregnancy, childbirth and the puerperium
	16	Certain conditions originating in the perinatal period
	17	Congenital malformations, deformations and chromosomal abnormalities
POISONING	18	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
	19	Injury, poisoning and certain other consequences of external causes
	20	External causes of morbidity and mortality
	21	Factors influencing health status and contact with health services
LIGHT REASON	22	Patient follow-up
	23	Medical consultation
	24	Blood donation
	25	Laboratory examination
	26	Unjustified absence
	27	Physiotherapy
	28	Dental consultation

289	36	33	239.554	30	1	2	1	2		

DATA PREPROCESCING

Cleaning the Reason for Absence

Reason for Absence	Date	Transportation Expense	Distance to Work	Age	Daily W
26	07/07/2015	289	36	33	
0	14/07/2015	118	13	50	
23	15/07/2015	179	51	38	
7	16/07/2015	279	5	39	
23	23/07/2015	289	36	33	
23	10/07/2015	179	51	38	
22	17/07/2015	361	52	28	
23	24/07/2015	260	50	36	
19	06/07/2015	155	12	34	
22	13/07/2015	235	11	37	

	Reason_1	Reason_2	Reason_3	Reason_4	Date
0	0	0	0	1	07/07/2015
1	0	0	0	0	14/07/2015
2	0	0	0	1	15/07/2015
3	1	0	0	0	16/07/2015
4	0	0	0	1	23/07/2015

1. Create a separate dataframe, containing dummy values for ALL available reasons

reason_columns = pd.get_dummies(df['Reason for Absence'], drop_first = True)

2. Split reason_columns into 4 types

reason_type_1 = reason_columns.loc[:,1:14].max(axis=1)
reason_type_2 = reason_columns.loc[:,15:17].max(axis=1)
reason_type_3 = reason_columns.loc[:,18:21].max(axis=1)
reason_type_4 = reason_columns.loc[:,22:].max(axis=1)

3. To avoid multicollinearity, drop the 'Reason for Absence' column from df

df = df.drop(['Reason for Absence'], axis = 1)

4. Concatenate df and the 4 types of reason for absence

df = pd.concat([df, reason_type_1, reason_type_2, reason_type_3, reason_type_4], axis = 1)

5. Assign names to the 4 reason type columns

```
column_names = ['Date', 'Transportation Expense', 'Distance to Work', 'Age',
                'Daily Work Load Average', 'Body Mass Index', 'Education', 'Children',
                'Pet', 'Absenteeism Time in Hours', 'Reason_1', 'Reason_2', 'Reason_3', 'Reason_4']
df.columns = column_names
```

6. Re-order the columns in df

```
column_names_reordered = ['Reason_1', 'Reason_2', 'Reason_3', 'Reason_4', 'Date', 'Transportation Expense',
                          'Distance to Work', 'Age', 'Daily Work Load Average', 'Body Mass Index', 'Education',
                          'Children', 'Pet', 'Absenteeism Time in Hours']
df = df[column_names_reordered]
```

DATA PREPROCESCING

Cleaning the Date

	Reason_1	Reason_2	Reason_3	Reason_4	Date	Transportation Expense	Distance to Work	Age
0	0	0	0	1	07/07/2015	289	36	33
1	0	0	0	0	14/07/2015	118	13	50
2	0	0	0	1	15/07/2015	179	51	38
3	1	0	0	0	16/07/2015	279	5	39
4	0	0	0	1	23/07/2015	289	36	33
5	0	0	0	1	10/07/2015	179	51	38
6	0	0	0	1	17/07/2015	361	52	28
7	0	0	0	1	24/07/2015	260	50	36
8	0	0	1	0	06/07/2015	155	12	34

	Reason_1	Reason_2	Reason_3	Reason_4	Month Value	Day of the Week	Transportation Expense	Distance to Work	Age
0	0	0	0	1	7	1	289	36	33
1	0	0	0	0	7	1	118	13	50
2	0	0	0	1	7	2	179	51	38
3	1	0	0	0	7	3	279	5	39
4	0	0	0	1	7	3	289	36	33
5	0	0	0	1	10	2	179	51	38
6	0	0	0	1	7	4	361	52	28
7	0	0	0	1	7	4	260	50	36
8	0	0	1	0	6	6	155	12	34

1. Convert the 'Date' column into datetime

```
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
```

2. Create a list with month values retrieved from the 'Date' column

```
list_months = []
for i in range(df.shape[0]):
    list_months.append(df['Date'][i].month)
```

3. Insert the values in a new column in df, called 'Month Value'

```
df['Month Value'] = list_months
```

4. Create a new feature called 'Day of the Week'

```
df['Day of the Week'] = df['Date'].apply(lambda x: x.weekday())
```

5. Drop the 'Date' column from df

```
df = df.drop(['Date'], axis = 1)
```

6. Re-order the columns in df

```
column_names_upd = ['Reason_1', 'Reason_2', 'Reason_3', 'Reason_4', 'Month Value', 'Day of the Week',
                    'Transportation Expense', 'Distance to Work', 'Age',
                    'Daily Work Load Average', 'Body Mass Index', 'Education', 'Children',
                    'Pet', 'Absenteeism Time in Hours']
df = df[column_names_upd]
```

DATA PREPROCESCING

Cleaning the ‘Education’

Education	Children	Pet	Absenteeism Time in Hours
1	2	1	4
1	1	0	0
1	0	0	2
1	2	0	4
1	2	1	2
1	0	0	2
1	1	4	8
1	4	0	4
1	2	0	40
3	1	1	8

Education	Children	Pet	Absenteeism Time in Hours
0	2	1	4
0	1	0	0
0	0	0	2
0	2	0	4
0	2	1	2
0	0	0	2
0	1	4	8
0	4	0	4
0	2	0	40
1	1	1	8

1	High school	538
2	Graduate	73
3	Postgraduate	40
4	A master or a doctor	4

1. map 'Education' variables; the result is a dummy

```
df['Education'] = df['Education'].map({1:0, 2:1, 3:1, 4:1})
```

0	High school	538
1	Not High School	117

2. Export the datagrams to .csv file, ready for machine learning

```
df.to_csv('Absenteeism-preprocessed.csv', index = False)
```

MACHINE LEARNING

Logistic Regression

Import the relevant libraries

```
# import the relevant libraries
import pandas as pd
import numpy as np
```

Load the data

```
# load the preprocessed CSV data
data_preprocessed = pd.read_csv('Absenteeism-preprocessed.csv')
```

```
# eyeball the data
data_preprocessed.head()
```

	Reason_1	Reason_2	Reason_3	Reason_4	Month Value	Day of the Week	Transportation Expense	Distance to Work	Age	Daily Work Load Average	Body Mass Index	Education	Children	Pet	Absenteeism Time in Hours
0	0	0	0	1	7	1	289	36	33	239.554	30	0	2	1	4
1	0	0	0	0	7	1	118	13	50	239.554	31	0	1	0	0
2	0	0	0	1	7	2	179	51	38	239.554	31	0	0	0	2
3	1	0	0	0	7	3	279	5	39	239.554	24	0	2	0	4
4	0	0	0	1	7	3	289	36	33	239.554	30	0	2	1	2

- Logistic Regression
  - indication of which variables are important for the analysis and which aren't.

MACHINE LEARNING

Creating Targets for Logistic Regression

Month Value	Day of the Week	Transportation Expense	Distance to Work	Age	Daily Work Load Average	Body Mass Index	Education	Children	Pet	Absenteeism Time in Hours
7	1	289	36	33	239.554	30	0	2	1	4
7	1	118	13	50	239.554	31	0	1	0	0
7	2	179	51	38	239.554	31	0	0	0	2
7	3	279	5	39	239.554	24	0	2	0	4
7	3	289	36	33	239.554	30	0	2	1	2

Month Value	Transportation Expense	Age	Body Mass Index	Education	Children	Pet	Excessive Absenteeism
7	289	33	30	0	2	1	1
7	118	50	31	0	1	0	0
7	179	38	31	0	0	0	0
7	279	39	24	0	2	0	1
7	289	33	30	0	2	1	0

1. **Find the median of 'Absenteeism Time in Hours'**

```
data_preprocessed['Absenteeism Time in Hours'].median()
```

2. **Parameterized code**

```
targets = np.where(data_preprocessed['Absenteeism Time in Hours'] >
    data_preprocessed['Absenteeism Time in Hours'].median(), 1, 0)
```

What are these classes?

Moderately Absent
(<= 3 hours)
TARGET: 0

Excessively Absent
(>= 4 hours)
TARGET: 1

In supervised learning we call these 0s & 1s are targets

3. **Create a checkpoint by dropping the unnecessary variables, also drop the variables we 'eliminated' after exploring the weights**

```
data_with_targets = data_preprocessed.drop(['Absenteeism Time in Hours','Day
of the Week', 'Daily Work Load Average','Distance to Work'],axis=1)
```

MACHINE LEARNING

Select the Inputs for the Regression

	Reason_1	Reason_2	Reason_3	Reason_4	Month Value	Transportation Expense	Age	Body Mass Index	Education	Children	Pet	Excessive Absenteeism
0	0	0	0	1	7	289	33	30	0	2	1	1
1	0	0	0	0	7	118	50	31	0	1	0	0
2	0	0	0	1	7	179	38	31	0	0	0	0
3	1	0	0	0	7	279	39	24	0	2	0	1
4	0	0	0	1	7	289	33	30	0	2	1	0

1. **# Create a variable that will contain the inputs (everything without the targets)**

```
unscaled_inputs = data_with_targets.iloc[:,:-1]
```

MACHINE LEARNING

Standardize the Data

	Reason_1	Reason_2	Reason_3	Reason_4	Month Value	Transportation Expense	Age	Body Mass Index	Education	Children	Pet
0	0	0	0	1	0.030796	1.005844	-0.536062	0.767431	0	0.880469	0.268487
1	0	0	0	0	0.030796	-1.574681	2.130803	1.002633	0	-0.019280	-0.589690
2	0	0	0	1	0.030796	-0.654143	0.248310	1.002633	0	-0.919030	-0.589690
3	1	0	0	0	0.030796	0.854936	0.405184	-0.643782	0	0.880469	-0.589690
4	0	0	0	1	0.030796	1.005844	-0.536062	0.767431	0	0.880469	0.268487
...	...	...	...	...	...	...	...	...	...	...	...
695	1	0	0	0	-0.568019	-0.654143	0.562059	-1.114186	1	0.880469	-0.589690
696	1	0	0	0	-0.568019	0.040034	-1.320435	-0.643782	0	-0.019280	1.126663
697	1	0	0	0	-0.568019	1.624567	-1.320435	-0.408580	1	-0.919030	-0.589690
698	0	0	0	1	-0.568019	0.190942	-0.692937	-0.408580	1	-0.919030	-0.589690
699	0	0	0	1	-0.568019	1.036026	0.562059	-0.408580	0	-0.019280	0.268487

1. **Define scaler as an object**

```
from sklearn.preprocessing import StandardScaler
absenteeism_scaler = StandardScaler()
```

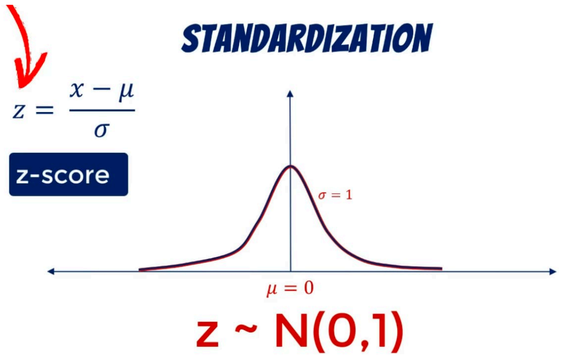
2. **Fit the data (calculate mean and standard deviation); they are automatically stored inside the object**

```
absenteeism_scaler.fit(unscaled_inputs)
```

3. **The scaled_inputs are now an ndarray, because sklearn works with ndarrays**

```
scaled_inputs = absenteeism_scaler.transform(unscaled_inputs)
```

"Data standardization is about making sure that data is internally consistent; that is, each data type has the same content and format"



MACHINE LEARNING

Split the Data Into Train & Test and Shuffle

Import the relevant module

```
# import train_test_split so we can split our data into train and test
from sklearn.model_selection import train_test_split
```

Split

```
# check how this method works
train_test_split(scaled_inputs, targets)
```

```
[     Reason_1  Reason_2  Reason_3  Reason_4  Month Value  \
 453         0         0         0         1    -0.268611
 277         0         0         0         0     0.629611
 597         0         0         0         1    -1.466241
 465         0         0         0         1     0.030796
 573         0         0         0         1    -0.568019
 ..        ...       ...       ...       ...          ...
 292         0         0         0         1    -0.268611
 61          0         0         0         1     0.629611
 390         0         0         1         0     0.030796
 305         0         0         0         1     0.929019
 601         0         0         0         1    -1.466241

      Transportation Expense       Age  Body Mass Index  Education  Children  \
 453                0.040034 -1.320435        -0.643782          0 -0.019280
 277                1.036026 -0.692937        -0.878984          0 -0.919030
 597               -0.654143  0.248310         1.002633          0 -0.919030
 465               -1.574681  0.091435         0.297027          0 -0.919030
 573               -0.654143  0.562059        -1.114186          1  0.880469
 ..                      ...       ...              ...        ...       ...
```

```
# declare 4 variables for the split
x_train, x_test, y_train, y_test = train_test_split(scaled_inputs, targets, #train_size = 0.8,
                                                    test_size = 0.2, random_state = 20)
```

```
# check the shape of the train inputs and targets
print (x_train.shape, y_train.shape)
```

```
(560, 11) (560,)
```

```
# check the shape of the test inputs and targets
print (x_test.shape, y_test.shape)
```

```
(140, 11) (140,)
```

MACHINE LEARNING

Fitting Model & Assessing its Accuracy

```
# import the LogReg model from sklearn
from sklearn.linear_model import LogisticRegression

# import the 'metrics' module, which includes important metrics we may want to use
from sklearn import metrics
```

Training the model

```
# create a logistic regression object
reg = LogisticRegression()
```

```
# fit our train inputs
# that is basically the whole training part of the machine learning
reg.fit(x_train,y_train)
```

```
/Users/zacharyvunguyen/opt/anaconda3/envs/zachary2019/lib/python3.7/site-packages/s
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solv
  FutureWarning)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# assess the train accuracy of the model
reg.score(x_train,y_train)
```

```
0.775
```

- **0.775**
  - Base on the data we used, the model learned to classify ~80% of the observation correctly

MACHINE LEARNING

Interpreting the Coefficients

Finding the intercept and coefficients

```
reg.intercept_
```

```
reg.coef_
```

```
unscaled_inputs.columns.values
```

```
feature_name = unscaled_inputs.columns.values
```

```
summary_table = pd.DataFrame (columns=['Feature name'], data = feature_name)

summary_table['Coefficient'] = np.transpose(reg.coef_)

summary_table
```

```
summary_table.index = summary_table.index + 1
summary_table.loc[0] = ['Intercept', reg.intercept_[0]]
summary_table = summary_table.sort_index()
summary_table
```

Interpreting the coefficients

```
summary_table['Odds_ratio'] = np.exp(summary_table.Coefficient)
```

```
summary_table
```

```
summary_table.sort_values('Odds_ratio', ascending=False)
```

	Feature name	Coefficient	Odds_ratio
3	Reason_3	2.940787	18.930743
1	Reason_1	2.602372	13.495716
2	Reason_2	0.843500	2.324489
4	Reason_4	0.637234	1.891243
6	Transportation Expense	0.619534	1.858062
10	Children	0.351950	1.421838
8	Body Mass Index	0.284103	1.328570
5	Month Value	0.005651	1.005667
7	Age	-0.176355	0.838320
9	Education	-0.263725	0.768185
11	Pet	-0.273698	0.760562
0	Intercept	-1.431381	0.238979

- **A feature is not particularly important if:**
  - Its coefficients is around 0
  - Its odds ratio is around 1

MACHINE LEARNING

Testing & Saving the Model

Testing the model

```
# assess the test accuracy of the model
reg.score(x_test,y_test)
```

```
0.7357142857142858
```

Based on the data that the model has NEVER seen before, in ~73.6% of the cases, the model will predict correctly if the person is going to be excessively absent

Save the model

```
# import the relevant module
import pickle
```

```
# pickle the model file
with open('model', 'wb') as file:
    pickle.dump(reg, file)
```

```
# pickle the scaler file
with open('scaler','wb') as file:
    pickle.dump(absenteeism_scaler, file)
```

pickle: a python module used to convert a python object into a character stream

MACHINE LEARNING

Preparing the Deployment of the Model Through a Module

absenteeism_module.py

```python
# import all libraries needed
import numpy as np
import pandas as pd
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.base import BaseEstimator, TransformerMixin

# the custom scaler class
class CustomScaler(BaseEstimator,TransformerMixin):

    def __init__(self,columns,copy=True,with_mean=True,with_std=True):
        self.scaler = StandardScaler(copy,with_mean,with_std)
        self.columns = columns
        self.mean_ = None
        self.var_ = None

    def fit(self, X, y=None):
        self.scaler.fit(X[self.columns], y)
        self.mean_ = np.array(np.mean(X[self.columns]))
        self.var_ = np.array(np.var(X[self.columns]))
        return self

    def transform(self, X, y=None, copy=None):
        init_col_order = X.columns
        X_scaled = pd.DataFrame(self.scaler.transform(X[self.columns]), columns=self.columns)
        X_not_scaled = X.loc[:,~X.columns.isin(self.columns)]
        return pd.concat([X_not_scaled, X_scaled], axis=1)[init_col_order]
```

```python
# create the special class that we are going to use from here on to predict new data
class absenteeism_model():

        def __init__(self, model_file, scaler_file):
            # read the 'model' and 'scaler' files which were saved
            with open('model','rb') as model_file, open('scaler', 'rb') as scaler_file:
                self.reg = pickle.load(model_file)
                self.scaler = pickle.load(scaler_file)
                self.data = None

        # take a data file (*.csv) and preprocess it in the same way as in the lectures
        def load_and_clean_data(self, data_file):

            # import the data
            df = pd.read_csv(data_file,delimiter=',')
            # store the data in a new variable for later use
            self.df_with_predictions = df.copy()
            # drop the 'ID' column
            df = df.drop(['ID'], axis = 1)
            # to preserve the code we've created in the previous section, we will add a column with 'NaN' strings
            df['Absenteeism Time in Hours'] = 'NaN'

            # create a separate dataframe, containing dummy values for ALL available reasons
            reason_columns = pd.get_dummies(df['Reason for Absence'], drop_first = True)

            # split reason_columns into 4 types
            reason_type_1 = reason_columns.loc[:,1:14].max(axis=1)
            reason_type_2 = reason_columns.loc[:,15:17].max(axis=1)
            reason_type_3 = reason_columns.loc[:,18:21].max(axis=1)
            reason_type_4 = reason_columns.loc[:,22:].max(axis=1)

            # to avoid multicollinearity, drop the 'Reason for Absence' column from df
            df = df.drop(['Reason for Absence'], axis = 1)

            # concatenate df and the 4 types of reason for absence
            df = pd.concat([df, reason_type_1, reason_type_2, reason_type_3, reason_type_4], axis = 1)

            # assign names to the 4 reason type columns
            # note: there is a more universal version of this code, however the following will best suit our current purposes

            column_names = ['Date', 'Transportation Expense', 'Distance to Work', 'Age',
                            'Daily Work Load Average', 'Body Mass Index', 'Education', 'Children',
                            'Pet', 'Absenteeism Time in Hours', 'Reason_1', 'Reason_2', 'Reason_3', 'Reason_4']
            df.columns = column_names

            # re-order the columns in df
            column_names_reordered = ['Reason_1', 'Reason_2', 'Reason_3', 'Reason_4', 'Date', 'Transportation Expense',
                                      'Distance to Work', 'Age', 'Daily Work Load Average', 'Body Mass Index', 'Education',
                                      'Children', 'Pet', 'Absenteeism Time in Hours']
            df = df[column_names_reordered]
```

MACHINE LEARNING

Preparing the Deployment of the Model Through a Module (Cont.)

absenteeism_module.py

```
# convert the 'Date' column into datetime
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')

# create a list with month values retrieved from the 'Date' column
list_months = []
for i in range(df.shape[0]):
    list_months.append(df['Date'][i].month)

# insert the values in a new column in df, called 'Month Value'
df['Month Value'] = list_months

# create a new feature called 'Day of the Week'
df['Day of the Week'] = df['Date'].apply(lambda x: x.weekday())


# drop the 'Date' column from df
df = df.drop(['Date'], axis = 1)

# re-order the columns in df
column_names_upd = ['Reason_1', 'Reason_2', 'Reason_3', 'Reason_4', 'Month Value', 'Day of the Week',
                    'Transportation Expense', 'Distance to Work', 'Age',
                    'Daily Work Load Average', 'Body Mass Index', 'Education', 'Children',
                    'Pet', 'Absenteeism Time in Hours']
df = df[column_names_upd]


# map 'Education' variables; the result is a dummy
df['Education'] = df['Education'].map({1:0, 2:1, 3:1, 4:1})

# replace the NaN values
df = df.fillna(value=0)

# drop the original absenteeism time
df = df.drop(['Absenteeism Time in Hours'],axis=1)

# drop the variables we decide we don't need
df = df.drop(['Day of the Week','Daily Work Load Average','Distance to Work'],axis=1)

# we have included this line of code if you want to call the 'preprocessed data'
self.preprocessed_data = df.copy()

# we need this line so we can use it in the next functions
self.data = self.scaler.transform(df)
```

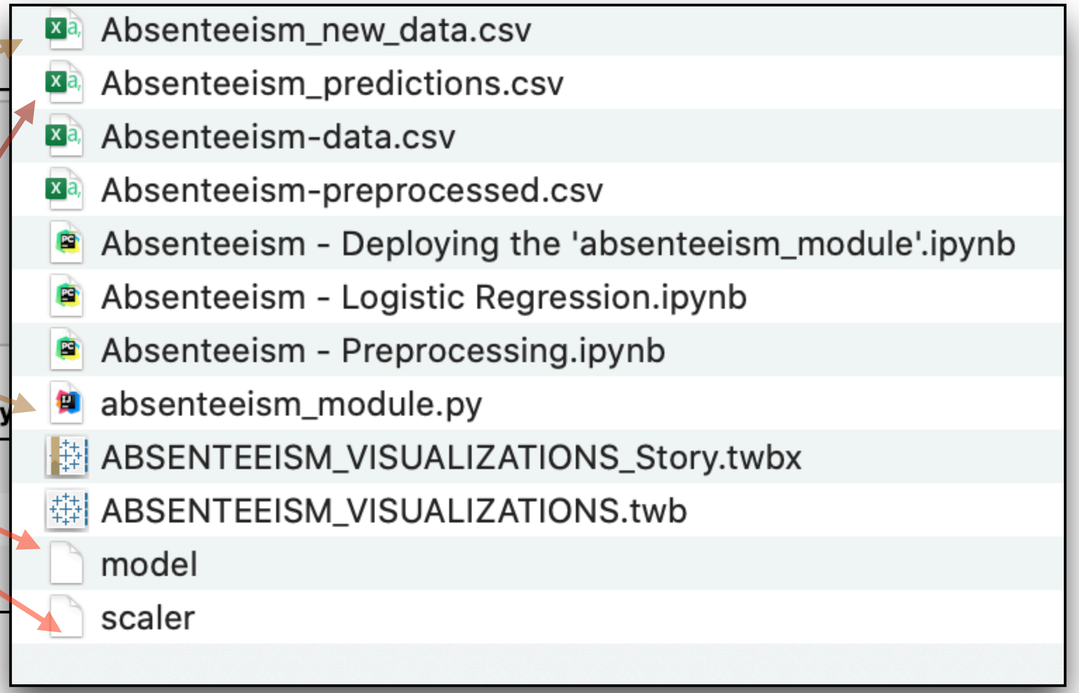
```
# a function which outputs the probability of a data point to be 1
def predicted_probability(self):
    if (self.data is not None):
        pred = self.reg.predict_proba(self.data)[:,1]
        return pred

# a function which outputs 0 or 1 based on our model
def predicted_output_category(self):
    if (self.data is not None):
        pred_outputs = self.reg.predict(self.data)
        return pred_outputs

# predict the outputs and the probabilities and
# add columns with these values at the end of the new data
def predicted_outputs(self):
    if (self.data is not None):
        self.preprocessed_data['Probability'] = self.reg.predict_proba(self.data)[:,1]
        self.preprocessed_data ['Prediction'] = self.reg.predict(self.data)
        return self.preprocessed_data
```

LOADING THE ‘ABESENTEEISM_MODULE’

Preparing the Deployment of the Model Through a Module (Cont.)

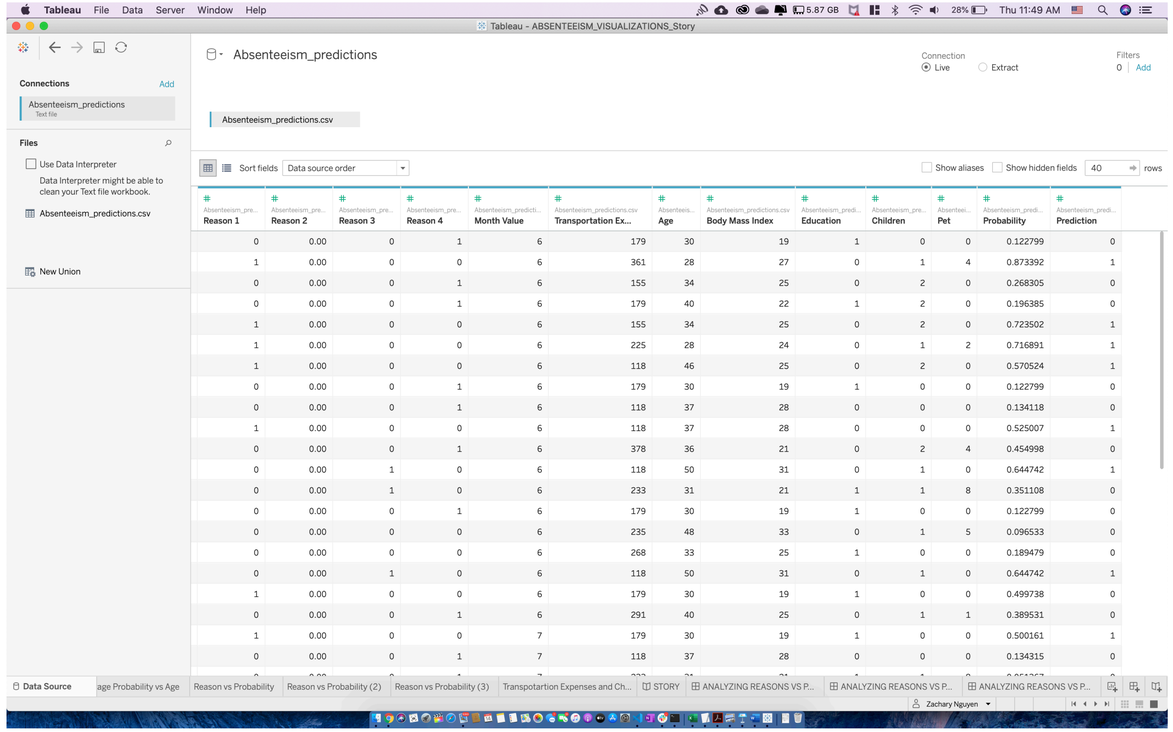
```
from absenteeism_module import *
model = absenteeism_model('model', 'scaler')

model.load_and_clean_data('Absenteeism_new_data.csv')

model.predicted_outputs()
```

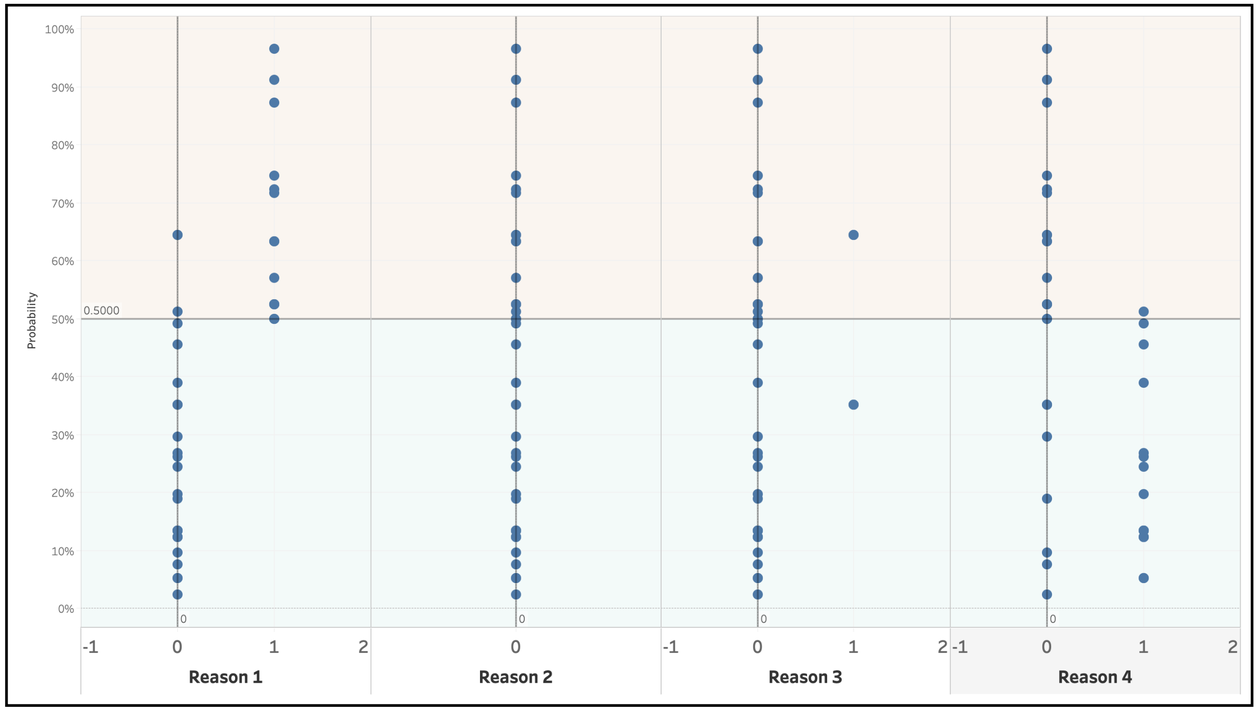
	Reason_1	Reason_2	Reason_3	Reason_4	Month Value	Transportation Expense	Age	Body
0	0	0.0	0	1	6	179	30	
1	1	0.0	0	0	6	361	28	
2	0	0.0	0	1	6	155	34	



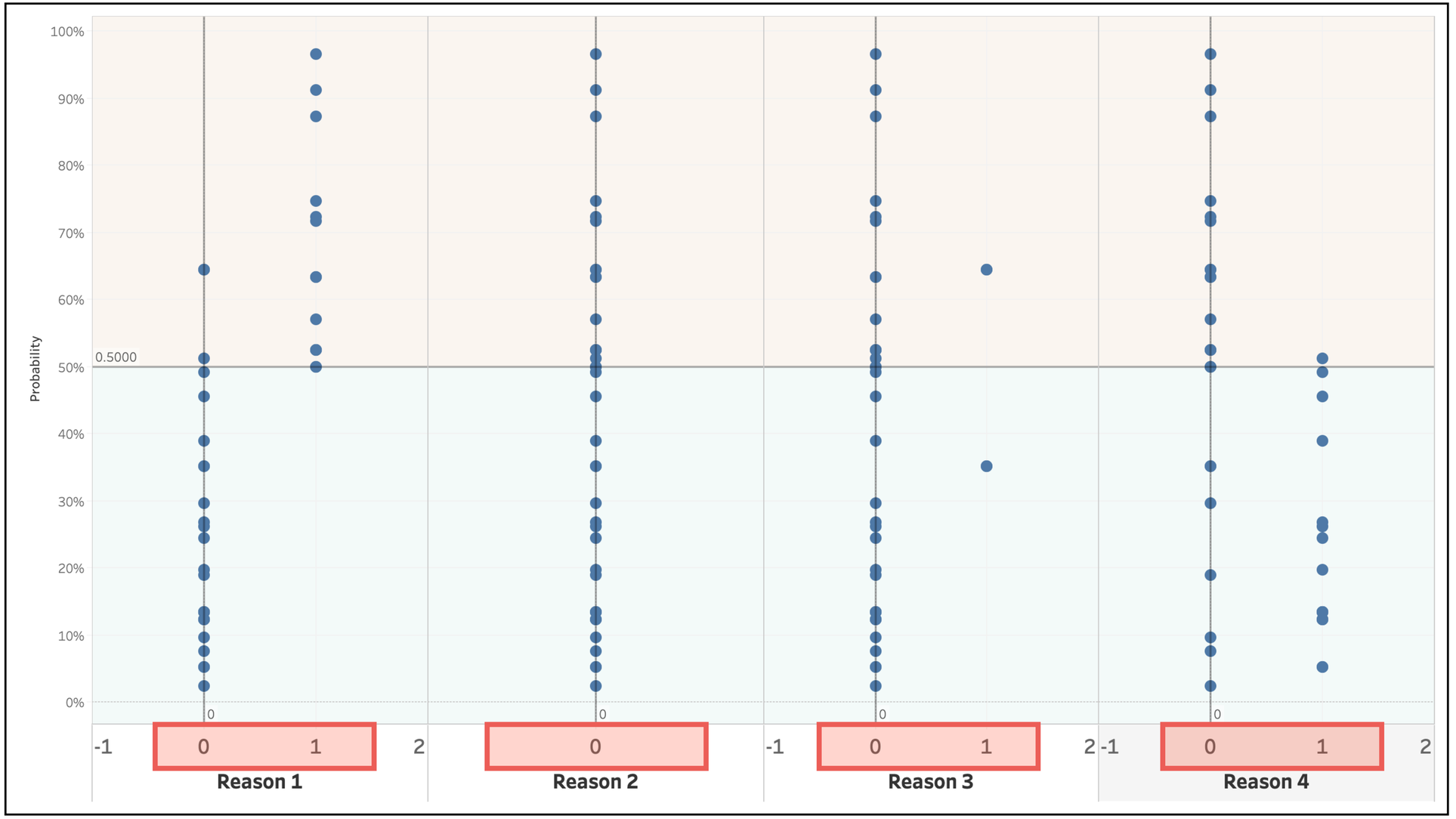

```
model.predicted_outputs().to_csv('Absenteeism_predictions.csv', index = False)
```

ANALYZING REASON VS PROBABILITY - TABLEAU



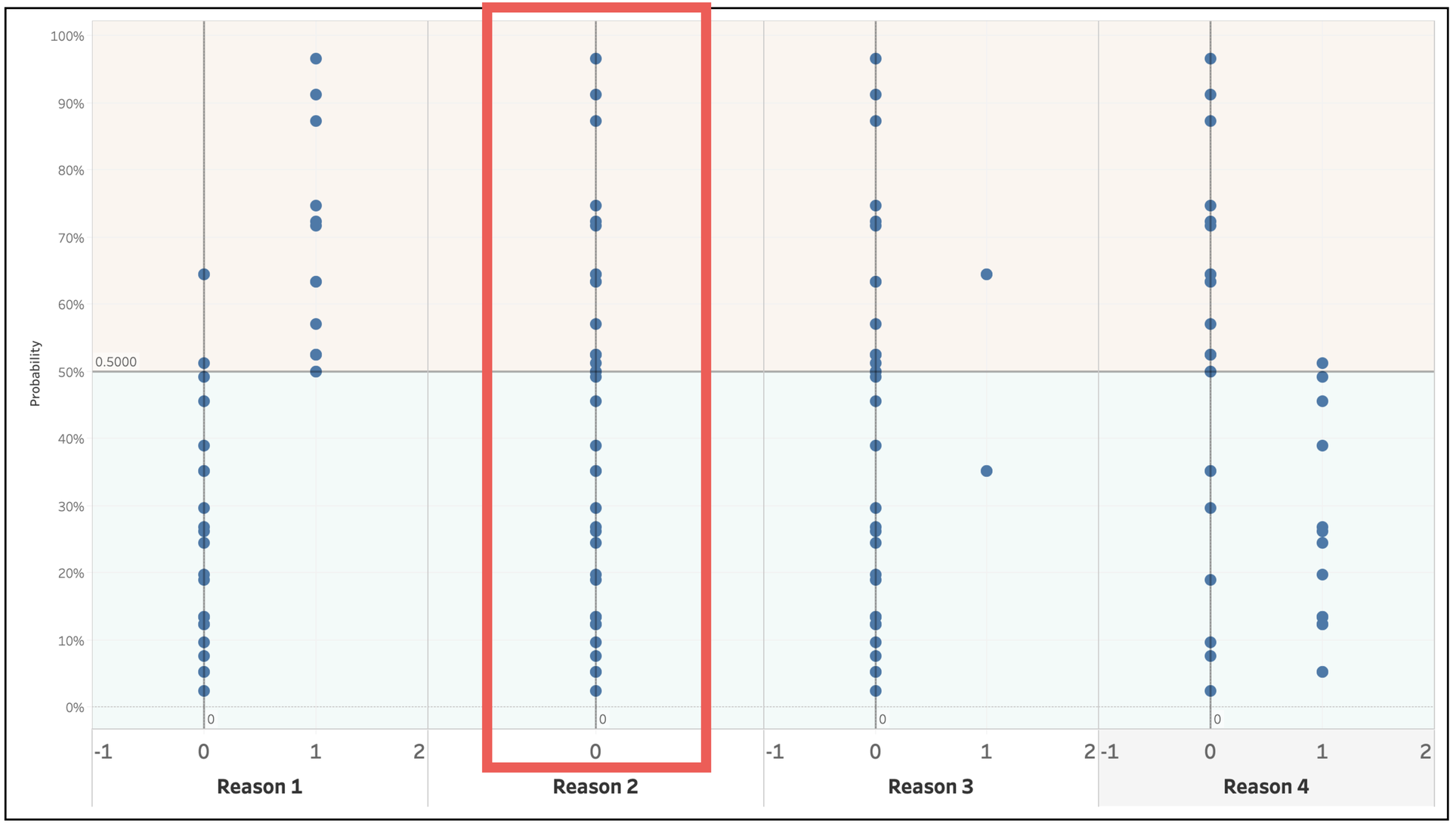


ANALYZING REASON VS PROBABILITY - TABLEAU



0: Absent
1: Not Absent

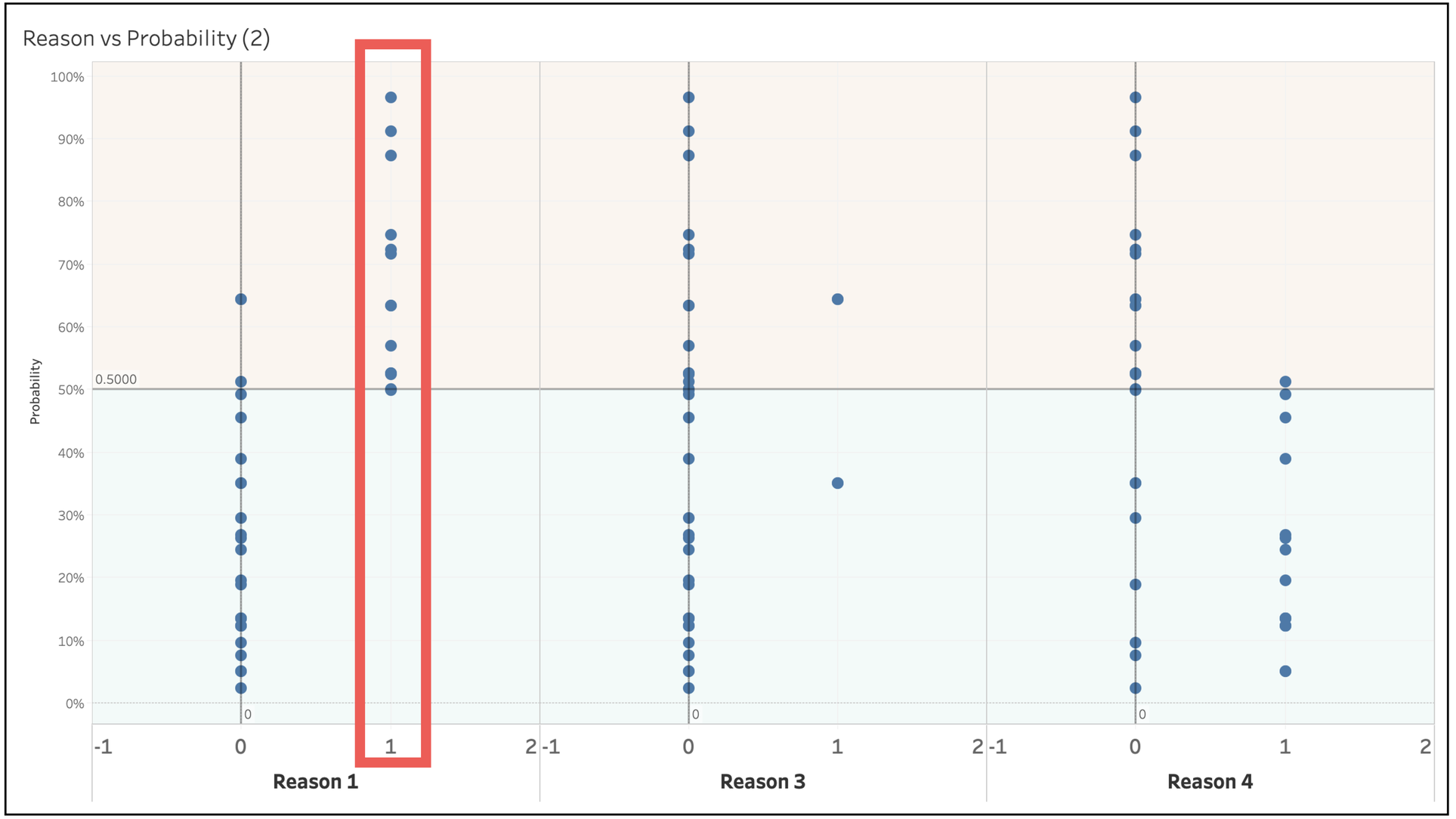
ANALYZING REASON VS PROBABILITY - TABLEAU



Reason 2:

None of our 40 observations has been away from work because of reason 2

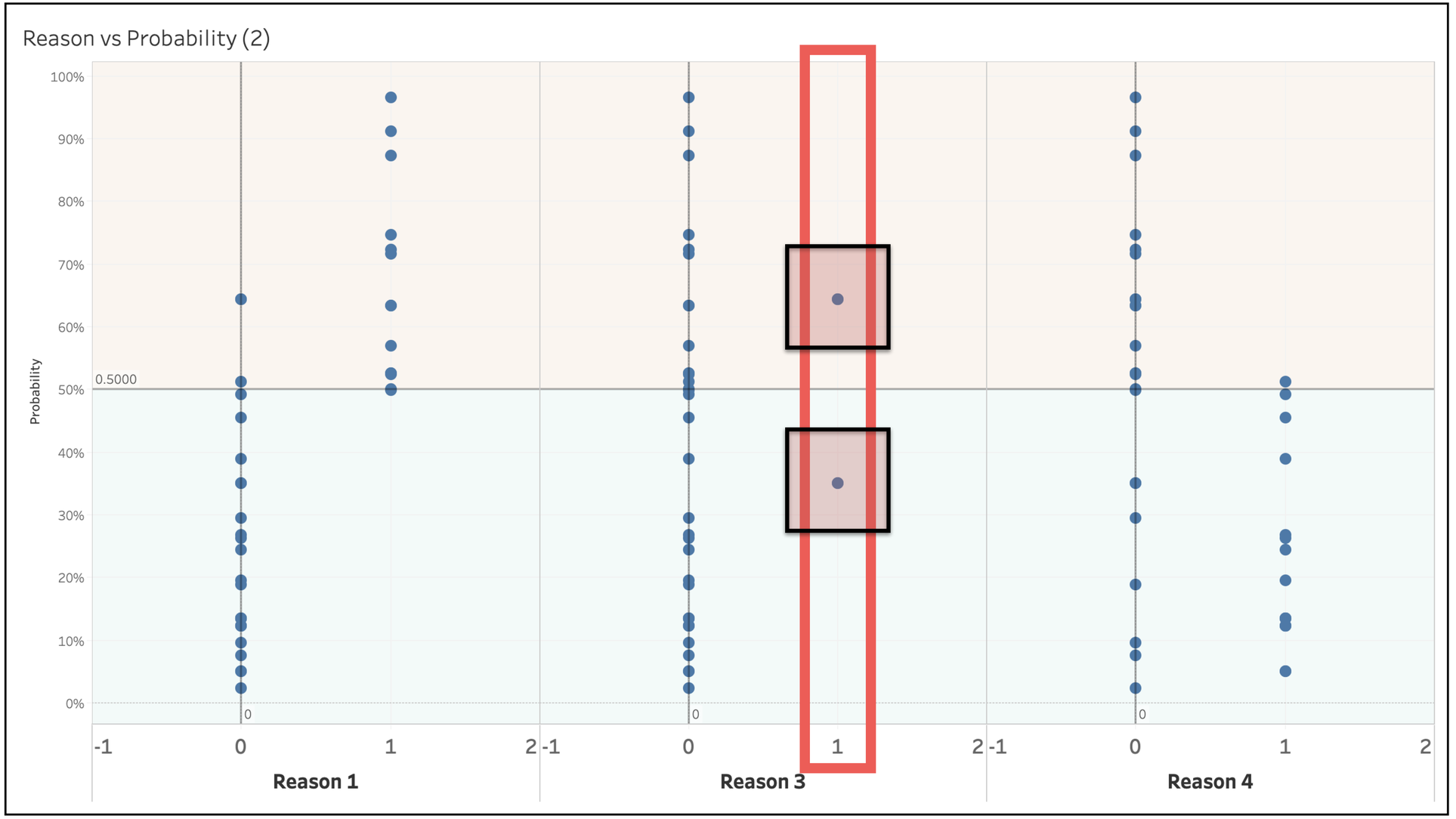
ANALYZING REASON VS PROBABILITY - TABLEAU



Reason 1:

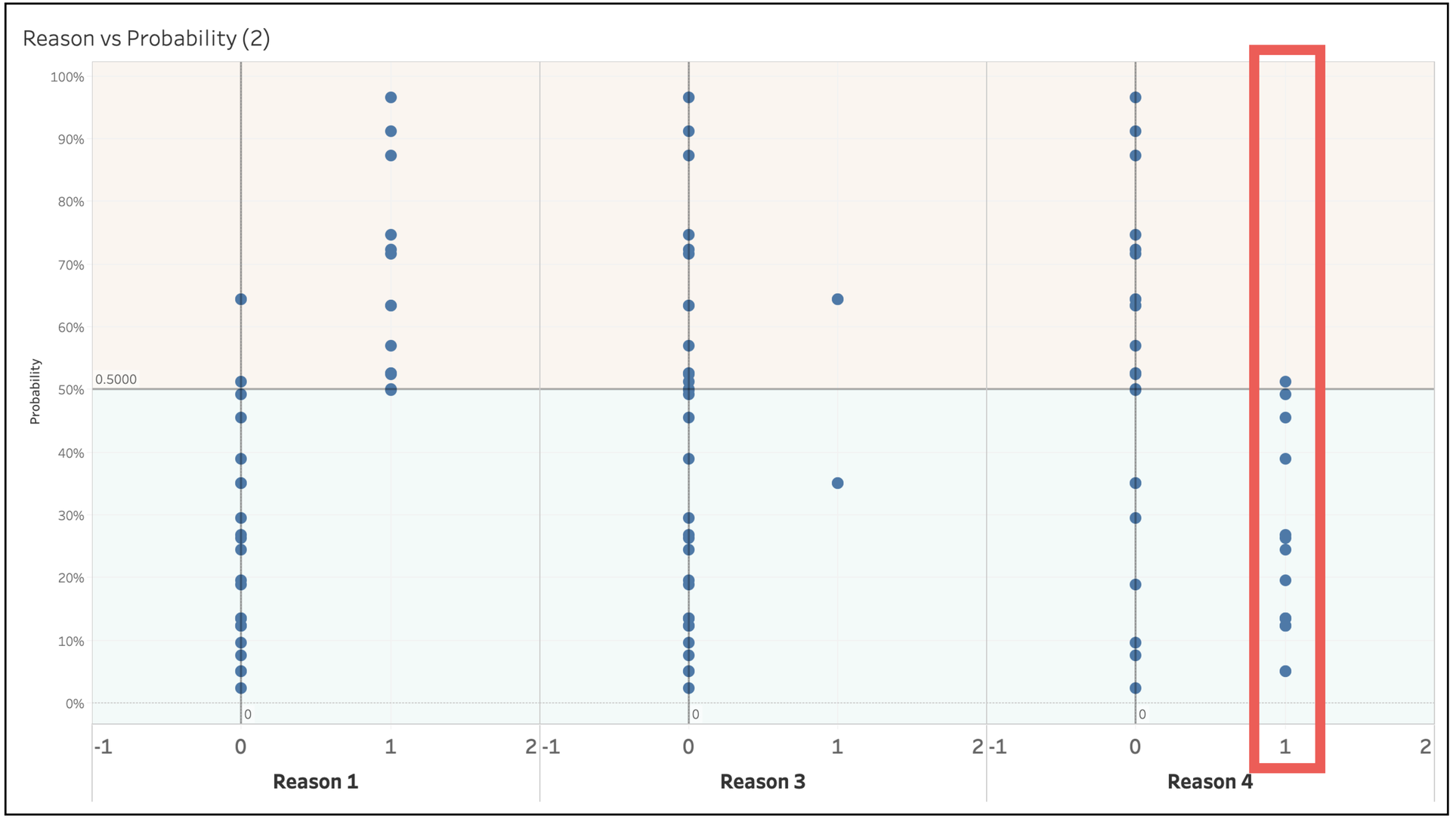
If an individual is support to be absent, he/she is expected to be away from work for ***more*** than three hours

ANALYZING REASON VS PROBABILITY - TABLEAU



Reason 3:
Doesn't tell us much about what to expect from individuals being absent because of a reason from Group 3.

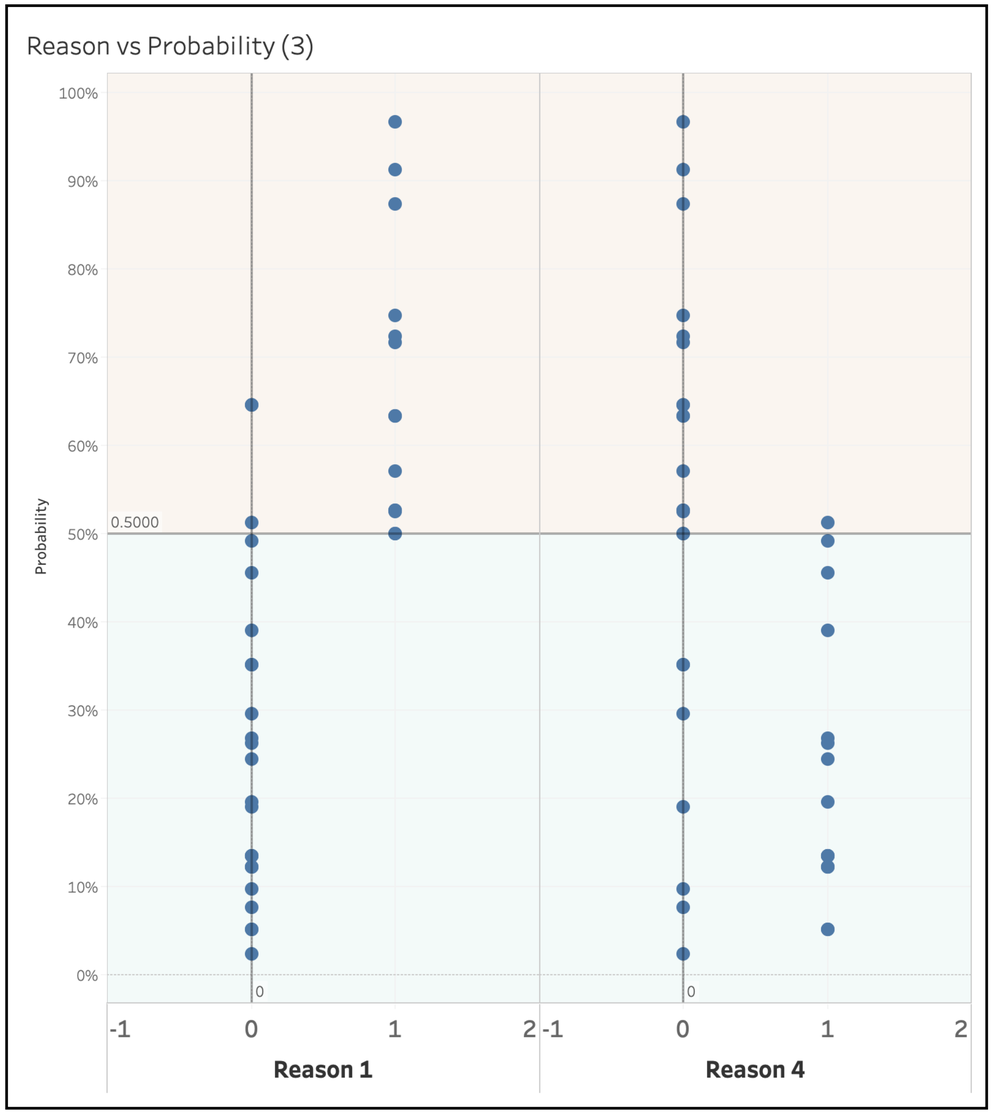
ANALYZING REASON VS PROBABILITY - TABLEAU



Reason 4:

If an individual is support to be absent, he/she is expected to be away from work for ***less*** than three hours

ANALYZING REASON VS PROBABILITY - TABLEAU

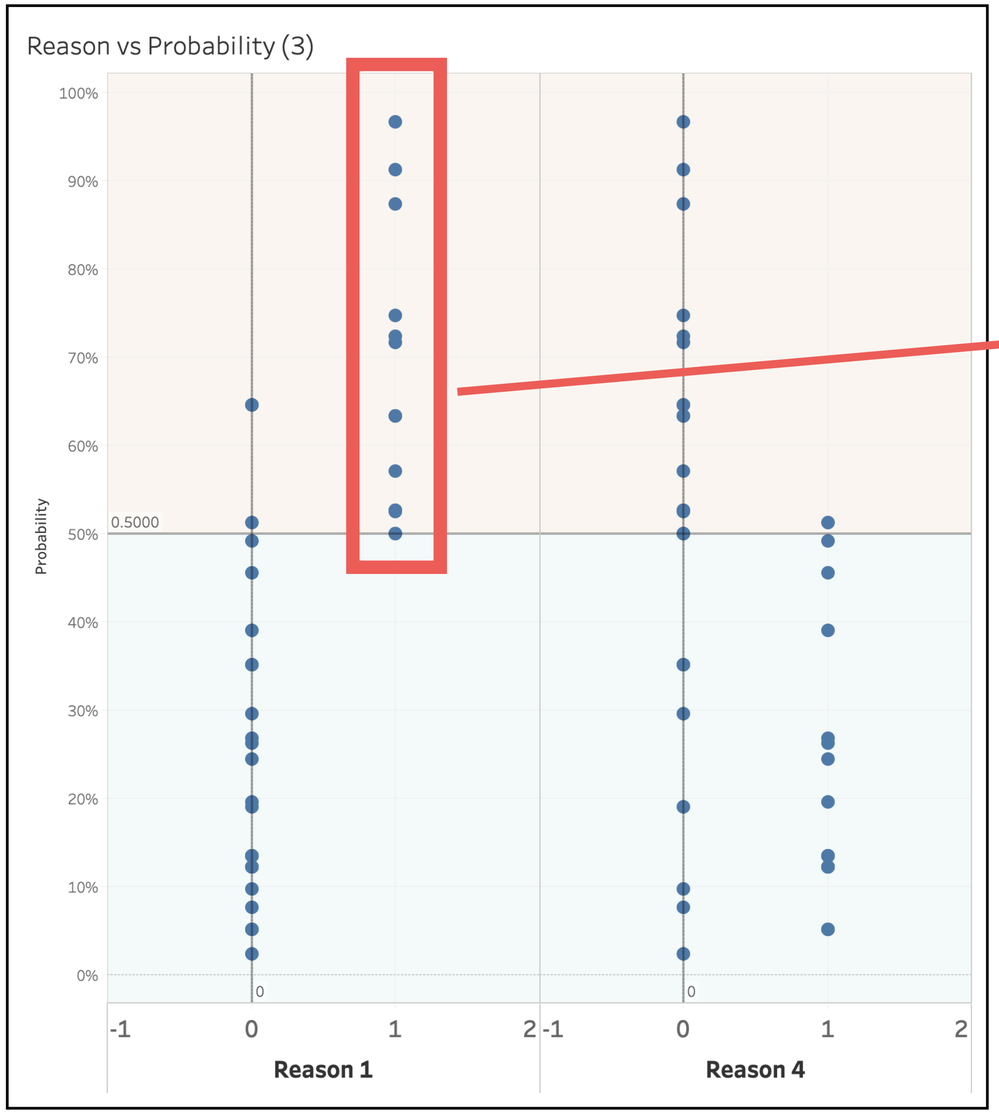


Do our numerical inferences match any business logic?

Can we have a more intuitive explanation?

Qualitative interpretation

ANALYZING REASON VS PROBABILITY - TABLEAU

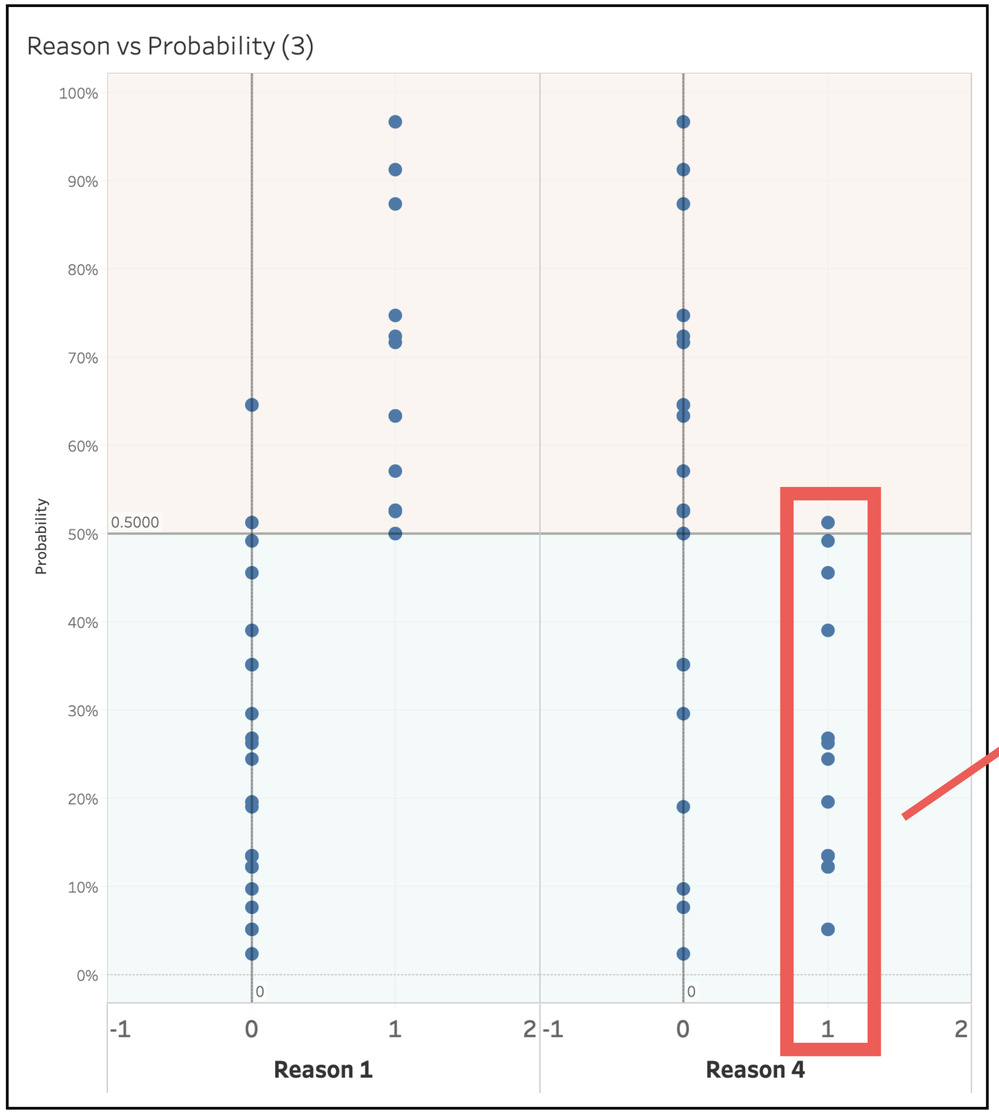


Serious Diseases!

1	Certain infectious and parasitic diseases
2	Neoplasms
3	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
4	Endocrine, nutritional and metabolic diseases
5	Mental and behavioral disorders
6	Diseases of the nervous system
7	Diseases of the eye and adnexa
8	Diseases of the ear and mastoid process
9	Diseases of the circulatory system
10	Diseases of the respiratory system
11	Diseases of the digestive system
12	Diseases of the skin and subcutaneous tissue
13	Diseases of the musculoskeletal system and connective tissue
14	Diseases of the genitourinary system

Good match between qualitative and quantitative!

ANALYZING REASON VS PROBABILITY - TABLEAU



Light Reasons

LIGHT REASON		
	22	Patient follow-up
	23	Medical consultation
	24	Blood donation
	25	Laboratory examination
	26	Unjustified absence
	27	Physiotherapy
	28	Dental consultation

Good match between qualitative and quantitative!

THE END

Thank you!