In [14]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

- Linear Regression: classification
- math background: Logic function and Sigmoid function

$$y = m * x + b$$

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$

In [2]:
```
df = pd.read_csv('https://raw.githubusercontent.com/codebasics/py/master/ML
```

In [4]:
```
df.head(2)
```

Out[4]:

	age	bought_insurance
0	22	0
1	25	0

In [97]:
```
# split train, test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df[['age']],df.bought_insu
```

In [98]:
```
y_test
```

Out[98]:
```
22    1
19    0
21    0
Name: bought_insurance, dtype: int64
```

In [99]:
```
from sklearn.linear_model import LogisticRegression
```

In [100]:
```
Lin_reg = LogisticRegression()
```

In [101]:
```
Lin_reg.fit(X_train,y_train)
```

Out[101]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [102]:
```
Lin_reg.predict(X_test)
```

Out[102]:
```
array([1, 0, 0])
```

In [103]:
```
# look at score
lin_reg.score(X_test,y_test)
```

Out[103]:
```
0.6065872792412028
```

In [104]:
```
Lin_reg.predict_proba(X_test)
```

Out[104]:
```
array([[0.48296773, 0.51703227],
       [0.9413583 , 0.0586417 ],
       [0.85090082, 0.14909918]])
```

In [65]:
```
y_test
```

Out[65]:
```
11    0
3     0
9     1
21    0
12    0
0     0
15    1
4     1
19    0
Name: bought_insurance, dtype: int64
```

In [59]:
```
X_train
```

Out[59]:

	age
23	45
10	18
22	40
14	49
16	25
2	47
26	23
18	19
6	55
17	58
5	56
1	25
24	50
20	21
25	54
8	62
7	60
13	29

- Exercise

In [138]:
```
df1 = pd.read_csv('https://raw.githubusercontent.com/codebasics/py/master/M
```

In [139]:
```
df1.head(2)
```

Out[139]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	W
0	0.38	0.53	2	157	3	
1	0.80	0.86	5	262	6	

In [140]:
```
# should replace with salary
import pandas as pd
```

In [141]:
```
df_merge = pd.concat([df1,pd.get_dummies(df1.salary)],axis='columns').drop(
```

In [176]:
```
df_merge = pd.concat([df_merge,pd.get_dummies(df1.Department)],axis = 'colu
```

In [177]:
```
df_merge
```

Out[177]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_compan
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	
...	...	...	...	...	
14994	0.40	0.57	2	151	
14995	0.37	0.48	2	160	
14996	0.37	0.53	2	143	
14997	0.11	0.96	6	280	
14998	0.37	0.52	2	158	

14999 rows × 31 columns

- what variable has direct or indirect influence on leave or continue to work

In [142]:
```
corr = df1.corr()
corr.style.background_gradient(cmap='coolwarm')
```

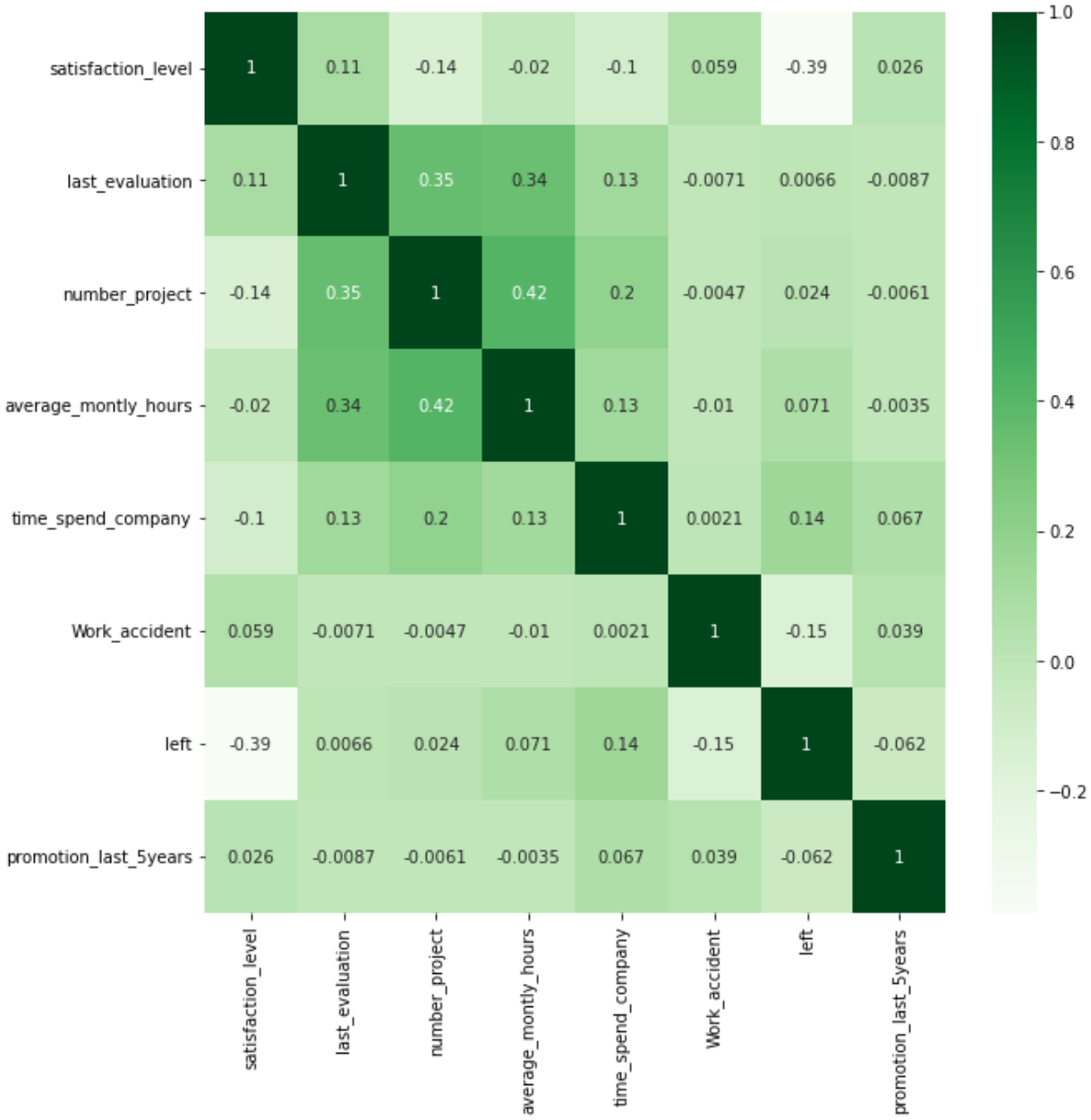
Out[142]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time
satisfaction_level	1.000000	0.105021	-0.142970	-0.020048	
last_evaluation	0.105021	1.000000	0.349333	0.339742	
number_project	-0.142970	0.349333	1.000000	0.417211	
average_montly_hours	-0.020048	0.339742	0.417211	1.000000	
time_spend_company	-0.100866	0.131591	0.196786	0.127755	
Work_accident	0.058697	-0.007104	-0.004741	-0.010143	
left	-0.388375	0.006567	0.023787	0.071287	
promotion_last_5years	0.025605	-0.008684	-0.006064	-0.003544	

In [143]:
```
import seaborn as sns
```

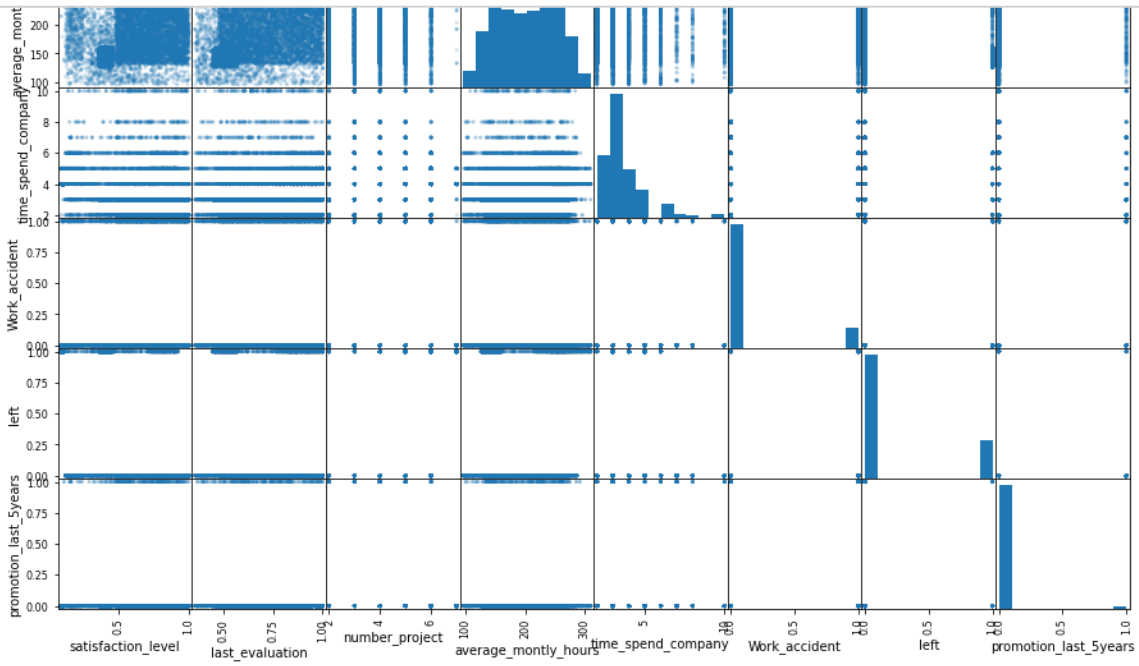
In [144]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr, cmap="Greens",annot=True)
```

Out[144]: <AxesSubplot:>



- Above, satisfaction_level is direct influence

In [121]:
```
pd.plotting.scatter_matrix(df1, alpha=0.2,figsize=(15,15))
```

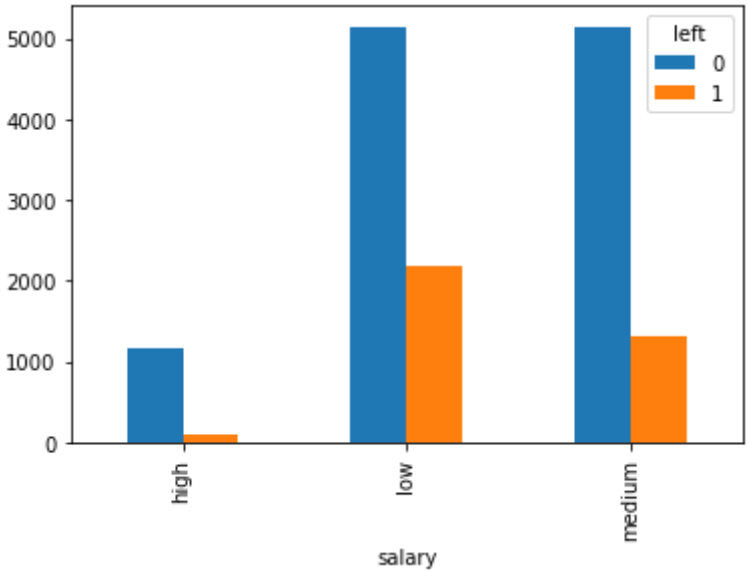


In [127]:
```
# show corr between salaries on rention
```

- pd.crosstab Compute a simple cross tabulation of two (or more) factors.

In [156]:
```
pd.crosstab(df1.salary,df1.left).plot(kind='bar')
```

Out[156]: <AxesSubplot:xlabel='salary'>



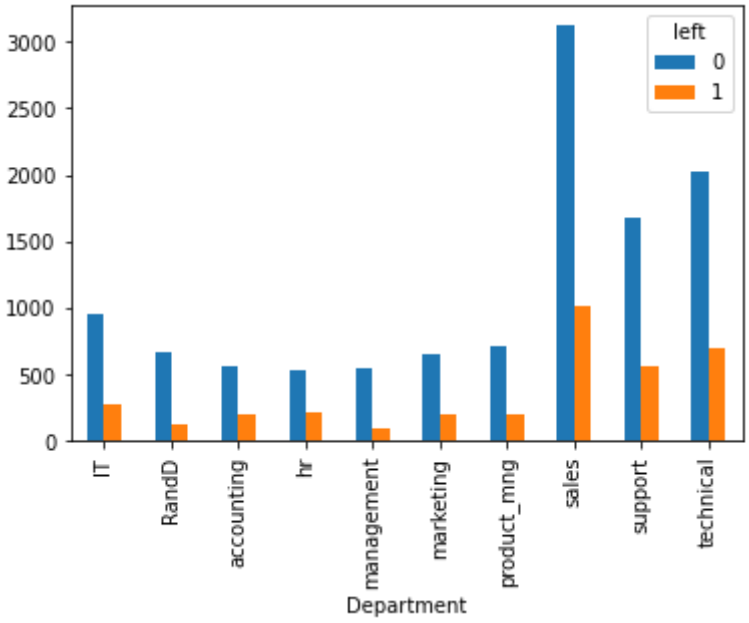
In [154]:
```
df1.groupby('salary')['left'].value_counts().plot(kind='bar',color = 'green
```

Out[154]: <AxesSubplot:xlabel='salary,left'>

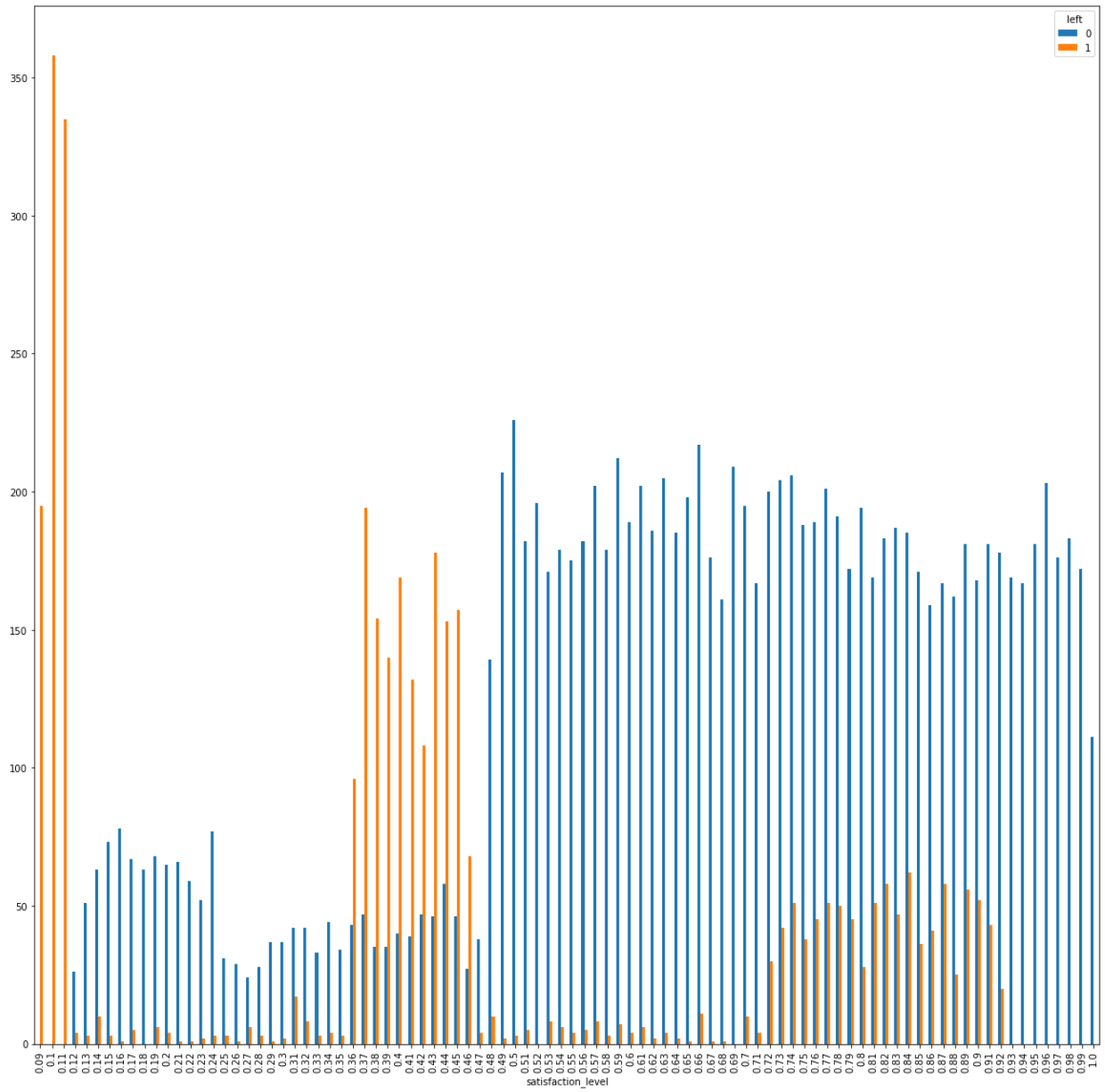
In [ ]:
```
# # show corr between department on rention
```

In [157]:
```
pd.crosstab(df1.Department,df1.left).plot(kind='bar')
```

Out[157]: <AxesSubplot:xlabel='Department'>



In [167]:
```
plt_1 = pd.crosstab(df1.satisfaction_level,df1.left).plot(kind='bar',figsiz
```



In [168]:
```
# build model
```

In [217]:
```
features = df_final.loc[:,df_final.columns!='left']
target = df_final.left
```

In [215]:
```
df_final['left'] = df1.left
```

In [216]:
```
df_final
```

Out[216]:

	high	low	medium	satisfaction_level	time_spend_company	Work_accident	IT	RandD	acc
0	0	1	0	0.38	3	0	0	0	
1	0	0	1	0.80	6	0	0	0	
2	0	0	1	0.11	4	0	0	0	
3	0	1	0	0.72	5	0	0	0	
4	0	1	0	0.37	3	0	0	0	
...	...	...	...	...	...	...	...	...	
14994	0	1	0	0.40	3	0	0	0	
14995	0	1	0	0.37	3	0	0	0	
14996	0	1	0	0.37	3	0	0	0	
14997	0	1	0	0.11	4	0	0	0	
14998	0	1	0	0.37	3	0	0	0	

14999 rows × 17 columns

In [211]:
```
features
```

Out[211]:

	high	low	medium	satisfaction_level	time_spend_company	Work_accident	IT	RandD	acc
0	0	1	0	0.38	3	0	0	0	
1	0	0	1	0.80	6	0	0	0	
2	0	0	1	0.11	4	0	0	0	
3	0	1	0	0.72	5	0	0	0	
4	0	1	0	0.37	3	0	0	0	
...	...	...	...	...	...	...	...	...	
14994	0	1	0	0.40	3	0	0	0	
14995	0	1	0	0.37	3	0	0	0	
14996	0	1	0	0.37	3	0	0	0	
14997	0	1	0	0.11	4	0	0	0	
14998	0	1	0	0.37	3	0	0	0	

14999 rows × 17 columns

In [203]:
```
df2 = pd.concat([pd.get_dummies(features['salary']),features],axis=1).drop(
df_final = pd.concat(
    [df2,pd.get_dummies(features['Department'])],axis=1).drop('Department',
```

In [204]:
```
df_final
```

Out[204]:

	high	low	medium	satisfaction_level	time_spend_company	Work_accident	IT	RandD	acc
0	0	1	0	0.38	3	0	0	0	
1	0	0	1	0.80	6	0	0	0	
2	0	0	1	0.11	4	0	0	0	
3	0	1	0	0.72	5	0	0	0	
4	0	1	0	0.37	3	0	0	0	
...	...	...	...	...	...	...	...	...	
14994	0	1	0	0.40	3	0	0	0	
14995	0	1	0	0.37	3	0	0	0	
14996	0	1	0	0.37	3	0	0	0	
14997	0	1	0	0.11	4	0	0	0	
14998	0	1	0	0.37	3	0	0	0	

14999 rows × 16 columns

In [218]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(features,target,test_size=
```

In [219]:
```
model = LinearRegression()
```

In [220]:
```
model.fit(X_train,y_train)
```

Out[220]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [221]:
```
model.predict(X_test)
```

Out[221]:
```
array([-0.03320312,  0.29467773,  0.62402344, ...,  0.33618164,
        0.39379883, -0.0090332 ])
```

In [222]:
```
model.score(X_train,y_train)
```

Out[222]:
```
0.19653719960330218
```

In [ ]:
```

```