Story About Dataset

McNeil(Plaintiff) injured in a Snowboard Accident.He contended that Snowboard he was using i.e Carve 3000 X5 was defective.Few Mock Jurors are assigned with task to find if the Carve 3000 snowboard X5 was really defective or not(i.e Liability) and assign damage percentages.

Plaintiff Argument:The holes that resist corosion are missing from the snowboard,which cause the accident.He is asking for 500,000 dollars to compensate his pain and sufferings.He asked for 490,000 dollars non-Economic damages for his pain and suffering and 10,000 dollars for economical damages.

Defedent Argument: Damages is high.

- There are 8 different paths.
- 1-4 path with no low Anchor
- 5-8 path with Low Anchor
  Limiting Jury instructions presented at 3,7,paths and Limiting Jury instruction with explanation introduced in 4,8 paths

Before a jury begins deliberatins, Judge will give instructions about the evidence of the case.For the fair trial,the court must sometimes limit the jurys considerations of a fact or evidence.This is done through a limiting instruction. Specifically,it tells the jury to disregard evidence completely or just for a specific purpose.

Mock Jurors were asked to watch videos where each video is of around 20-30 mins.They were asked 41 different questions(attributes of our dataset) which includes the information about Juror(like Age,sex,Education,Income etc) and questions like Was McNeil negligent?,If you find that Mc.Neil was fault,assign damage percentage, etc.Q40 and Q41 includes if there is any change in the decision if Limiting Jury instruction introduced.

There were two variations of the Plaintiff's closing argument.In both variations,Plaintiff made the same liability argument followed by one of the two damage demands.Plaintiff's attorney asked the Jury to award either 250,000 dollars or 5 million dollars to compensate for pain and suffering associated with the back injury,We viewed that 250,000 dollars as an objectively reasonable figure because it is roughly the average award given by mock jurors earlier.

Mock Jurors were then asked to render a decision on both liability and damages.These individual jurors results were then combined with 11 other randomly selected jurors decision to create a mock jury descision.

Objectives:

- Path 1-4 vs Path 5-8 Regression Analysis
- Regression Analysis for the Stair case and Snowboard Scenarios
- Linear model on Discounted Damages and the Path
- Damages calculations for each Path of new dataset
- Regression Analysis of Liability vs Path, Education and Income
- Plots for the data
- Regression Analysis for Q40 and Q41
- Damage calculations for the merged dataset

New Dataset(Snowboard) after cleaning and extracting required columns

- After Cleaning and filtering the data,we are left with 729 participants out of 804 participants.
- We have cleaned the data(cleaning.ipynb) and loaded cleaned data to cleaning.csv table)

In [1]:

```
import pandas as pd
data18 = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
data14 = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
```

For rest of the calculations, we will consider total 4 paths.so replacing path 5-8 with 1-4 respectively for data14.

In [2]:

```
data14['Path'].replace([5,6,7,8], [1,2,3,4], inplace = True)
```

In [3]:

```
data14.dtypes
```

Out[3]:

```
Unnamed: 0                                              int64
StartDate                                              object
EndDate                                                object
Duration                                                int64
Was_snowboard_sold_McNeil_defective_14                float64
Is_substantial_factor_McNeil_injuries_14              float64
Non_economic_damages_McNeil_suffered_14               float64
Was_McNeil_negligent                                  float64
McNeil_negligence_substantial_factor_for_injuries     float64
Percentage_of_responsibility_X5                       float64
Percentage_of_responsibility_McNeil                   float64
Was_snowboard_sold_McNeil_defective_58                float64
Is_substantial_factor_McNeil_injuries_58              float64
Economic_damages_McNeil_suffer_58                     float64
Non_economic_damages_McNeil_suffered_58               float64
Q40                                                   float64
Q41                                                    object
Path                                                    int64
Education                                               int64
Income                                                  int64
Total_perc                                            float64
Liability                                              object
is_substantial                                        float64
Liability_updated                                      object
Total_Damages                                         float64
Discounted_Damages                                    float64
dtype: object
```

In [4]:

```
data14.shape
```

Out[4]:

```
(729, 26)
```

1: Plot Of Total Damages(Discounted) vs Path.

Note: We used Violin Plot because it allows a deeper understanding of the density of distribution.

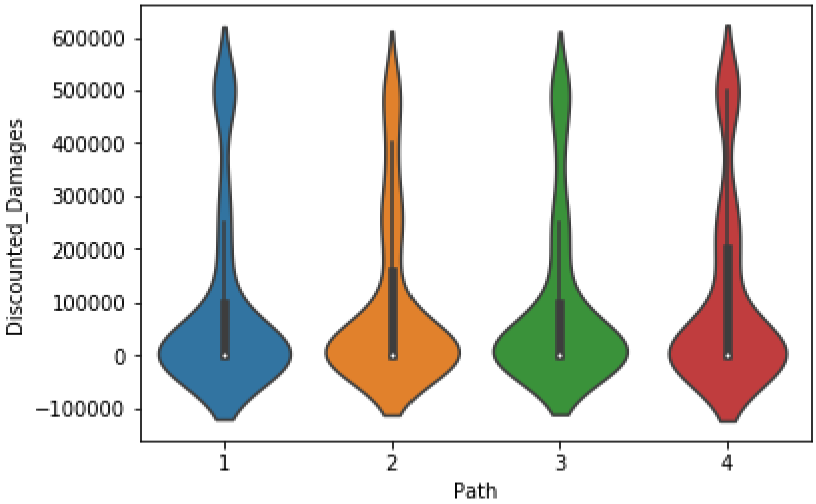
1) Plot including 0's for Total Damages(Discounted Damages)

In [5]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

_ = sns.factorplot(x='Path', y='Discounted_Damages', kind='box',data=data14, siz
e=6)
```

In [6]:

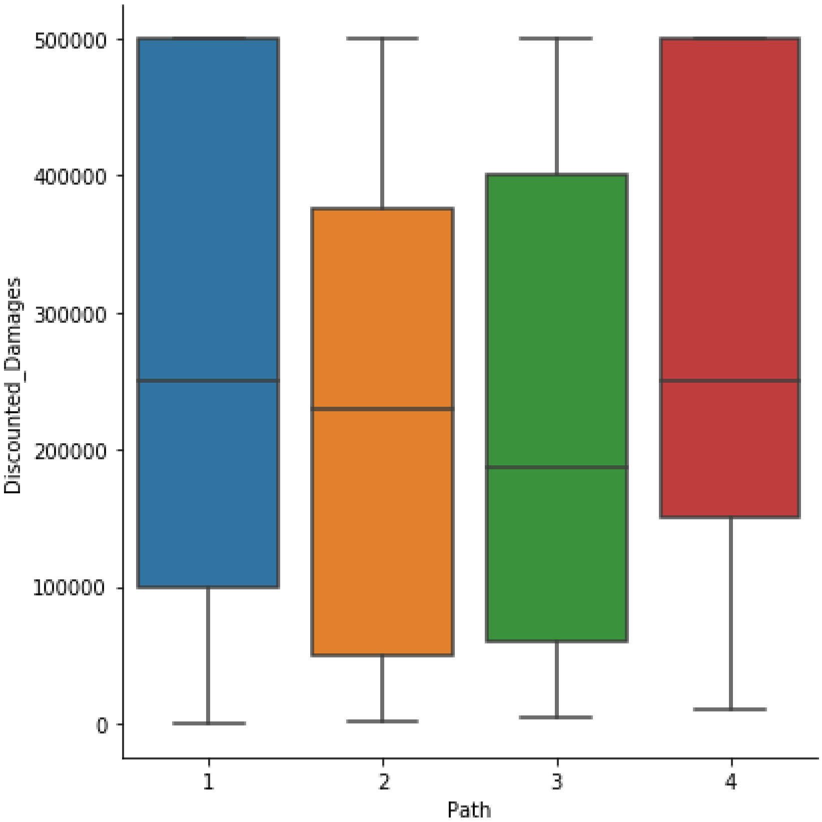
```
_ = sns.violinplot(x="Path", y="Discounted_Damages", data=data14, inner = 'box',
 size=6)
```

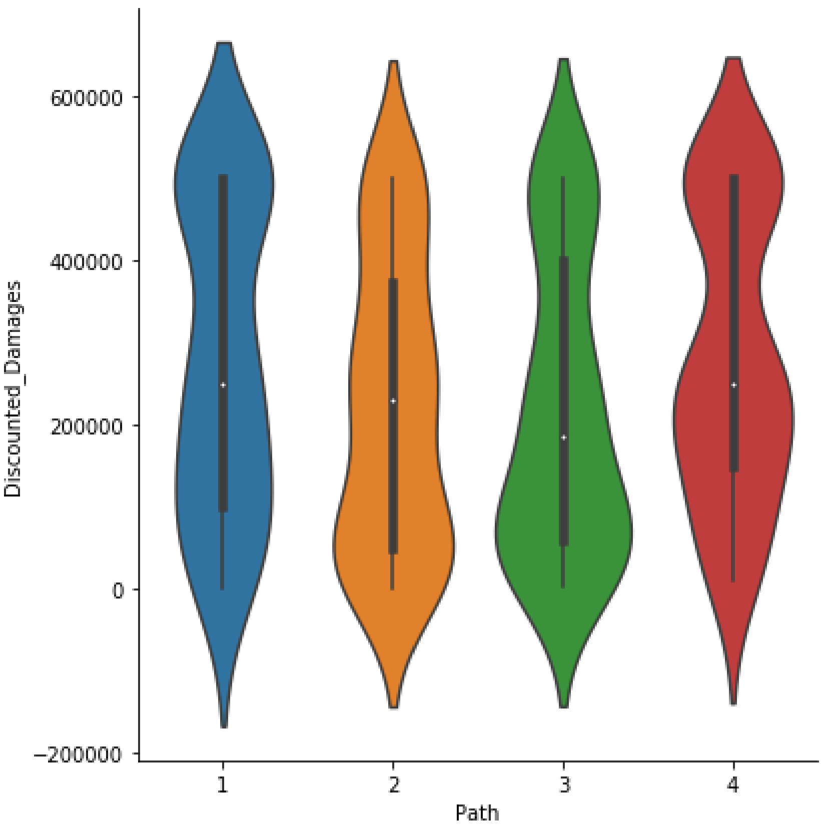


2) Box plot excluding 0s for Total Damages (Discounted Damages)

In [7]:

```
_ = sns.factorplot(x='Path', y='Discounted_Damages', kind='box',data=data14[data
14.Discounted_Damages >0], size=6)
_ = sns.factorplot(x='Path', y='Discounted_Damages', kind='violin',data=data14[d
ata14.Discounted_Damages >0], size=6)
```





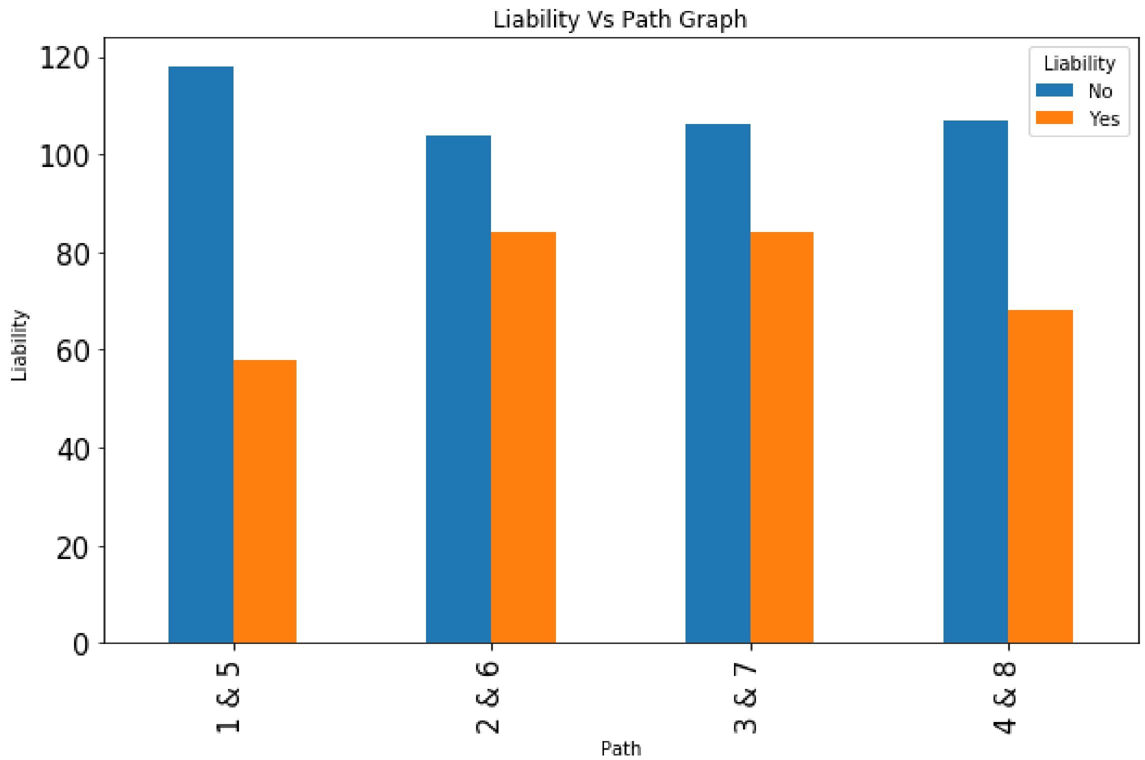
2: Plot of Liability vs Paths

In [8]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns


a = data14['Path']
a = a.astype(str)
a.replace(['1','2','3','4'],['1 & 5','2 & 6','3 & 7','4 & 8'],inplace = True)
b = data14['Liability']


pd.crosstab(a,b).plot(kind='bar', fontsize = 15, figsize=(10,6))
plt.title('Liability Vs Path Graph')
plt.xlabel('Path')
plt.ylabel('Liability')
plt.savefig('Juror Response vs Path')

pd.crosstab(a,b,margins=True, margins_name='Total')
```

Out[8]:

Liability	No	Yes	Total
Path			
1 & 5	118	58	176
2 & 6	104	84	188
3 & 7	106	84	190
4 & 8	107	68	175
Total	435	294	729



Finding the Winrate, Expected Damages, mean , median and SD for Discounted Damages

Here we are calculating the damages mean, median sd and win rate percentage when the plaintiff wins for path 1-4.

1) Case Expected Damage calculation

Adding Contributory negligence

Contributory negligence is when Juror find the Defendant Liable and also responded that the plaintiff is also somehow responsible for the accident i:e Liability == 'Yes' and Was_McNeil_negligent == 'Yes'

Was_McNeil_negligent: 1 (Yes) : 2 (No)

In [9]:

```
data14['Was_McNeil_negligent'] = data14['Was_McNeil_negligent'] .astype(str)
data14['Was_McNeil_negligent'].replace(['1.0', '2.0'], ['Yes','No'], inplace = True)
```

In [10]:

```
#data14.query("Was_McNeil_negligent == 'Yes' & Liability == 'Yes' & Path == 1")
a = data14['Path']
a = a.astype(str)
a.replace(['1','2','3','4'],['1 & 5','2 & 6','3 & 7','4 & 8'],inplace = True)
b = data14.query("Was_McNeil_negligent == 'Yes' & Liability == 'Yes'")
b = b.rename(columns={'Was_McNeil_negligent': 'Contributory_negligence'})
pd.crosstab(a,b.Contributory_negligence)
```

Out[10]:

Contributory_negligence	Yes
Path	
1 & 5	30
2 & 6	47
3 & 7	47
4 & 8	30

In [11]:

```
## Finding winrate percentage for each path
import numpy as np

ratedf=pd.DataFrame(data14[['Liability','Path','Was_McNeil_negligent']])
ratedf['winrate_percentage']=ratedf.Liability
ratedf['Discounted_damages_mean']=pd.to_numeric(data14.Discounted_Damages)
ratedf['Discounted_damages_median']=pd.to_numeric(data14.Discounted_Damages)
ratedf['Discounted_damages_sd']=pd.to_numeric(data14.Discounted_Damages)
ratedf['winrate_percentage'] = ratedf['winrate_percentage'].map({"Yes":1, "No":0
})
ratedf['No.of.Participants']=ratedf['Path']

winrate_damages_expected=ratedf.groupby('Path').aggregate(
    {'No.of.Participants':'count','winrate_percentage': np.mean
     ,'Discounted_damages_mean': np.mean
     ,'Discounted_damages_median':np.median
     ,'Discounted_damages_sd':np.std

    })
winrate_damages_expected['winrate_percentage']=winrate_damages_expected['winrate
_percentage']*100.0
winrate_damages_expected
```

Out[11]:

	No.of.Participants	winrate_percentage	Discounted_damages_mean	Discountec
Path				
1	176	32.954545	91850.795455	0.0
2	188	44.680851	98567.021277	0.0
3	190	44.210526	94718.421053	0.0
4	175	38.857143	108744.285714	0.0

Finding the Damages, mean , median and SD when plaintiff wins.

In [12]:

```
winrate_damages_plaintiffwin = ratedf.loc[(ratedf['Liability']=='Yes')].groupby(
'Path').aggregate(
    {'No.of.Participants':'count',
    'Discounted_damages_mean': np.mean
     ,'Discounted_damages_median':np.median
     ,'Discounted_damages_sd':np.std
    })
winrate_damages_plaintiffwin
```

Out[12]:

	No.of.Participants	Discounted_damages_mean	Discounted_damages_median
Path			
1	58	278719.655172	250000.0
2	84	220602.380952	230000.0
3	84	214244.047619	187500.0
4	68	279856.617647	250000.0

Question 1 :

 With respect to the first question, I realize that answers from participants in versions 1 and 5 are meaningless. They did not see evidence of added core inserts. As far as the analysis, I think we want to see if this answer predicted how people responded to the liability questions. For example, did people that said "Yes this evidence strongly suggested the Carve 3000 was defective” find liability more often than people that answered “No”.

Here Q40 is **"Did the fact that X5 added core inserts to the later Carve 3000 model, affect your view as to whether the original Carve 3000 was defective?"**

The Values are:

- 1 = Yes, it strongly suggested that the original Carve 3000 was defective.
- 2 = Yes, it somewhat suggested that the original Carve 3000 was defective.
- 3 = No, it did not suggest that the original Carve 3000 was defective.

So first lets check the brief summary table for each scenario.

In [13]:

```
newdf_2_8 = data18[~data18['Path'].isin([1,5])]
newdf_2_8['Q40'] = newdf_2_8['Q40'].astype(str)
a = newdf_2_8['Q40'].replace(['1.0','2.0','3.0'], ['Yes','Maybe','No'])
a = a[a.apply(len) > 0]
b = newdf_2_8['Liability']

p = pd.crosstab([b,a], newdf_2_8.Path,  margins=True , margins_name='Total')
p
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: Sett
ingWithCopyWarning:
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/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
```

Out[13]:

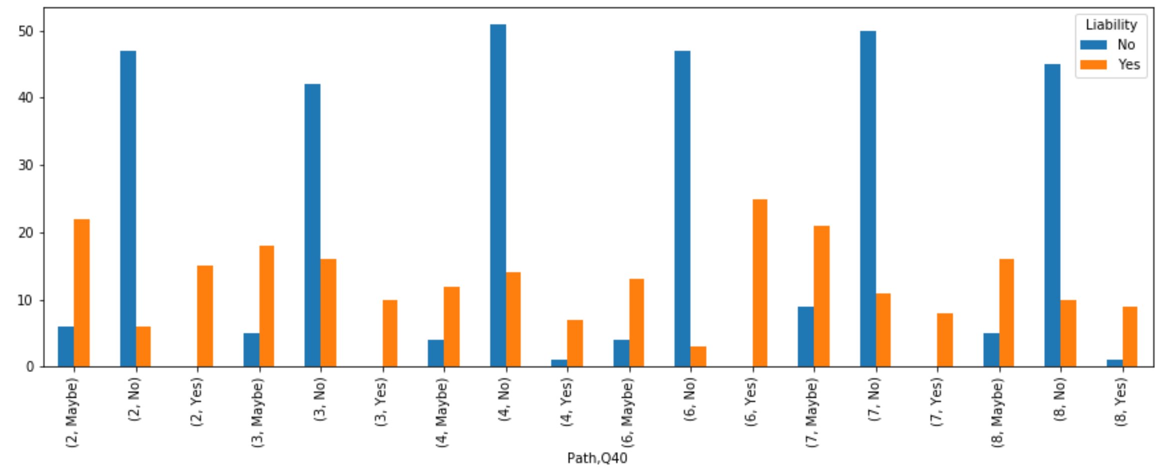
	Path	2	3	4	6	7	8	Total
Liability	**Q40**							
No	**Maybe**	6	5	4	4	9	5	33
	No	47	42	51	47	50	45	282
	Yes	0	0	1	0	0	1	2
Yes	**Maybe**	22	18	12	13	21	16	102
	No	6	16	14	3	11	10	60
	Yes	15	10	7	25	8	9	74
Total		96	91	89	92	99	86	553

Observation

Below is the plot for Path 2,3,4,6,7,8

In [14]:

```
_ = pd.crosstab([newdf_2_8.Path,a], b).plot(kind='bar', fontsize = 10, figsize=(
15,5))
```



Observation Based on the graph:

1) When a juror has strongly suggested that the original Carve 3000 was defective, all of them responded to the liability question as **"Yes"**.

2) When a juror is somewhat suggested that the original Carve 3000 was defective, they are more likely to respond the liability question as **"Yes"**.

3) When a juror is saying "No" to Q40 and saying that it does not suggest that the original Carve 3000 was defective, they are more likely to respond the liability question as **"No."**

.

Professor Bernard's comment:

One more **comment in the 1st question**. We also want to see if the jury instructions in 3,4 and 7,8 help participant resist the evidence.
So we want to also compare if verdicts as a function of yes, no, maybe in 2 and 5 are different than those in 3,4,7 & 8. If they do, we want to see if the different instructions matter. That is verdicts as function of yes, no, maybe or different in 3 and 7 vs 4 and 8.

1. Regression of Q40 vs Liability (Model 1)

Effect of juror response for Q40 (Yes, No, Maybe) on Liability.

In [15]:

```
import statsmodels.formula.api as smf # stats model formula
import seaborn as sb # statistical visulaization
%matplotlib inline
import matplotlib.pyplot as plt
sb.set(style="darkgrid", context="talk")
from scipy import stats

newdf_2_8.Liability =  newdf_2_8.Liability.astype('category')
newdf_2_8['Liability'] = data14['Liability'].map({"Yes":1, "No":0})
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:3643:
SettingWithCopyWarning:
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/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: Sett
ingWithCopyWarning:
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/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

In [16]:

```
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
logit_model = smf.logit(formula= 'Liability ~ C(Q40)', data = newdf_2_8.query("Q
40 == '2.0' |  Q40 == '3.0'")).fit()
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.490371
         Iterations 5
```

Out[16]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	477
Model:	Logit	**Df Residuals:**	475
Method:	MLE	**Df Model:**	1
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.2347
Time:	17:44:01	**Log-Likelihood:**	-233.91
converged:	True	**LL-Null:**	-305.65
		LLR p-value:	4.582e-33

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	1.1285	0.200	5.635	0.000	0.736	1.521
C(Q40)[T.3.0]	-2.6760	0.246	-10.896	0.000	-3.157	-2.195

We got convergence error when running the logistic regression model with Q40 values as 1(i:e Yes), as all people who said **Yes to Q40** also said **Yes to Liability**. so we removed value with 1 and rerun the model.

In [17]:

```
print(np.exp(1.8092+3.0961))  ## Odd when Q40 is maybe
print(np.exp(12.7067 -13.9936))   ## Odd when Q40 is No
```

```
135.0034049537509
0.27612544656126825
```

Observation From Model 1

The model can be written as :

$$Liability = \beta_0 + \beta_1 Q40(No)$$

> The intercept (β_0) is the **Base condition** when the Juror said "No" to Q40 i:e do not suggest that the original Carve 3000 was defective find X5 as Liable.

From model coefficient, we can see that there is an increase in coefficient from "No" to "Maybe", that means low people awarded liability for "No" than "Maybe"

Interm of odds:

The odds of Juror saying "No" to Q40 and find X5 liable is **-3.0961**Interm of percentage its 21%.

The odds ratio of Juror saying "No" to Q40 and find X5 liable is **1.8092** .Interm of percentage it is 86%.

- Q40 Response(Yes) Liability (100%)
- Q40 Response (Maybe) Liability (85%)
- Q40 Response(No) Liability(26%)

 Professor Bernard comment:

 Now to lets check the 2nd part of the question i:e to compare if verdicts as a function of yes, no, maybe in 2 and 6 are different than those in 3,4,7 & 8. If they do, check if the different instructions matter. That is verdicts as function of yes, no, maybe or different in 3 and 7 vs 4 and 8..

To compare path 2,6 vs 3,7 vs 4,8. we have replaced the path 6,7,8 with 2,3,4.

In [18]:

```
newdf_2_8.Path.replace([6,7,8],[2,3,4], inplace = True)
p = newdf_2_8.Path
p = p.astype(str)
p = p.replace(['2','3','4'],['2 & 6','3 & 7','4 & 8'])
```

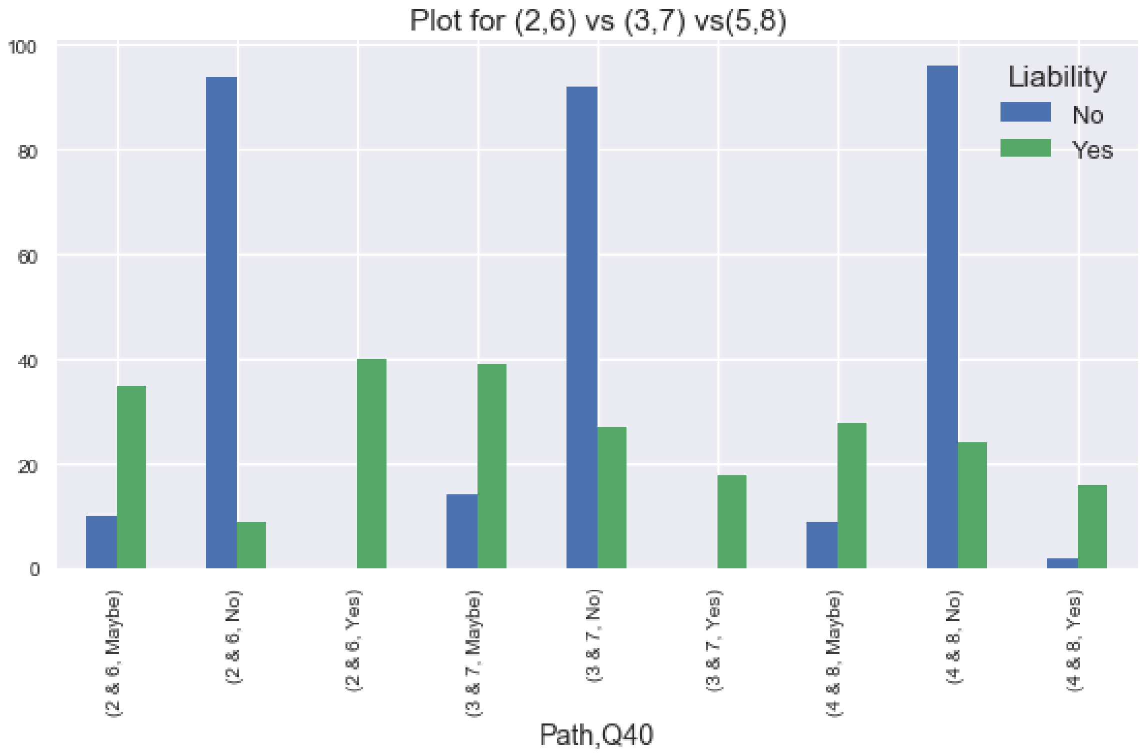
```
/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:4619:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(new_data)
```

Plot for Liability in path 2, 3 and 4.

So now let's see if these paths have any significant difference or not.

In [19]:

```
_ = pd.crosstab([p,a], b).plot(kind='bar', fontsize = 10,
                 title='Plot for (2,6) vs (3,7) vs(5,8)', figsize=(10,5))
```



In [20]:

```
newdf_2_8.Q40.dropna(inplace = True)
newdf_2_8.Q40.replace(['1.0', '2.0' , '3.0'],['Yes', 'May Be', 'No'], inplace =
True)
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/series.py:2993: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(result)
/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:4619:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(new_data)
```

In [21]:

```
pd.crosstab([newdf_2_8.Q40,data14.Liability],p, margins=True)
```

Out[21]:

	Path	2 & 6	3 & 7	4 & 8	All
Q40	Liability				
May Be	No	10	14	9	33
	Yes	35	39	28	102
No	No	94	92	96	282
	Yes	9	27	24	60
Yes	No	0	0	2	2
	Yes	40	18	16	74
All		188	190	175	553

Observations (Verdict rate for each path)

1: From Table:

For Path (2,6) : (No jury Instruction, Subsequent Remedial Mesure (Added code Insert))

> When a juror strongly accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **Yes to Q40, the verdict rate is 100 %.**
>
> When a juror somewhat accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **May Be to Q40, the verdict rate is 86 %.**
>
> When a juror is denied the fact that later core insert implies that the original Carve 3000 was defective, i:e **No to Q40, the verdict rate is 12%.**

For Path (3,7): (Simple Limiting Jury Instruction)

> When a juror strongly accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **Yes** to Q40, the verdict rate is **100 %.**
>
> When a juror somewhat accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **May Be** to Q40, the verdict rate is **85%.**
>
> When a juror is denied the fact that later core insert implies that the original Carve 3000 was defective, i:e **No** to Q40, the verdict rate is **27%.**

For Path (4,8): (Jury Instruction with Explanation)

> When a juror strongly accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **Yes** to Q40, the verdict rate is **100%.**
>
> When a juror somewhat accepts the fact that later core insert implies that the original Carve 3000 was defective, i:e **May Be** to Q40, the verdict rate is **86%.**
>
> When a juror is denied the fact that later core insert implies that the original Carve 3000 was defective, i:e **No** to Q40, the verdict rate is **24%.**

Winrate Observation from Table:

Overal Winrate for path (2,6) is 49% , for path (3,7) is 50% and path (3,7) is 45%.

1) From the table we can see that, the winrate slightly increasing(1%) when limiting jury instructions introduce in Path 3 and 7.

2) Win rate decreases by around 5% when complex jury instruction added in path 4 and 8 as compare to path 3 and 7 keeping the response fixed.

2: From Plot:

1) When a juror has **strongly** suggested that the original Carve 3000 was defective, all of them responded to the liability question as **"Yes"** for all paths.

2) When a juror is **somewhat** suggested that the original Carve 3000 was defective, they are more likely to respond the liability question as **"Yes"**.

3) When a juror is saying **"No"** to Q40 and saying that it does not suggest that the original Carve 3000 was defective, they are more likely to respond the liability question as **"No"**.

** We can say that Q40 is an important factor while deciding Liability**

2. Regression of Q40 vs Liability (Model 2)

Effect of juror response for Q40 (Yes, No, Maybe) for different paths on Liability.

Lets try to do the regression between Liability ~ Path+Q40. But before that, we should check if there is any collinearity between those attributes exists or not.

In [22]:

```
import statsmodels.formula.api as sm
results = sm.ols(formula= 'Liability ~ C(Path) + C(Q40)',data =newdf_2_8).fit()
results.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	Liability	**R-squared:**	0.439
Model:	OLS	**Adj. R-squared:**	0.435
Method:	Least Squares	**F-statistic:**	107.2
Date:	Thu, 28 Jun 2018	**Prob (F-statistic):**	2.09e-67
Time:	17:44:01	**Log-Likelihood:**	-235.59
No. Observations:	553	**AIC:**	481.2
Df Residuals:	548	**BIC:**	502.8
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7159	0.039	18.253	0.000	0.639	0.793
C(Path)[T.3]	0.0681	0.039	1.759	0.079	-0.008	0.144
C(Path)[T.4]	0.0471	0.039	1.194	0.233	-0.030	0.125
C(Q40)[T.No]	-0.5807	0.038	-15.313	0.000	-0.655	-0.506
C(Q40)[T.Yes]	0.2305	0.054	4.283	0.000	0.125	0.336

Omnibus:	37.513	**Durbin-Watson:**	2.041
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	47.770
Skew:	0.576	**Prob(JB):**	4.24e-11
Kurtosis:	3.863	**Cond. No.**	5.54

Condition number can be computed using eigenvalues of the normalized predictor variable. If the value is small it indicates that there is no collinearity between the variable else there should be an error indicating that the condition number is high.

Cond. No -> 5.54 which is less. Removing "Yes" due to convergence error.

In [23]:

```
logit_model = smf.logit(formula= 'Liability ~ C(Q40) + C(Path) ',
              data = newdf_2_8[newdf_2_8.Q40.isin(['No','May Be'])]).fit()
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.485939
         Iterations 6
```

Out[23]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	477
Model:	Logit	**Df Residuals:**	473
Method:	MLE	**Df Model:**	3
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.2417
Time:	17:44:02	**Log-Likelihood:**	-231.79
converged:	True	**LL-Null:**	-305.65
		LLR p-value:	8.163e-32

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	0.7916	0.257	3.083	0.002	0.288	1.295
C(Q40)[T.No]	-2.7236	0.251	-10.858	0.000	-3.215	-2.232
C(Path)[T.3]	0.5537	0.291	1.901	0.057	-0.017	1.125
C(Path)[T.4]	0.4937	0.301	1.642	0.101	-0.096	1.083

In [24]:

```
print(np.exp(1.4395)/(1+np.exp(1.4395)))
print(np.exp(1.4395+0.6312)/(1+np.exp(1.4395+0.6312)))
print(np.exp(0.6312 - 0.5524)/(1+np.exp(0.6312 - 0.5524)))
```

```
0.8083772116328087
0.8880225872818766
0.5196898124951566
```

Observation From Model 2

The model can be written as :

$$Liability = \beta_0 + \beta_1 Q40(No) + \beta_2 path(3) + \beta_4 Path(4)$$

> The intercept (β_0) is the **Base condition** when the path is 2 and Juror said "May be" to Q40 i:e a juror somewhat suggested that the original Carve 3000 was defective find X5 as Liable belong to path 2.

Interpretation of Liability for Path 2,6 vs 3,7:

As we have replaced the path 6,7 with 2,3 let's see the Liability for path 2 vs 3.

(β_0) : Log odd of finding Liability at Path 2 when juror what somewhat agree to Q40 and no limiting jury instruction introduced is **1.4395.**Interm of percentage around 81%.

(β_1) : When we change the path from 2 to 3 (when simple limiting jury instruction was introduced) , it tells us the log odd ratio of awarding Liability i:e **0.6312**

Keeping the juror response fixed for Q40, The odds of awarding Liability is about 2 times greater when **No** Limiting jury instruction introduced (path 2) than Limiting Jury instruction introduced (in path 3).

Comparing to the base scenario, keeping the response for Q40 fixed, when we change from scenario 2 to 3, the odds ratio of awarding liability is 0.6312, Interm of percentage 89%(8% increase) from the path (2-3).

Interpretation of Liability from Path 2,6 vs 4,8:

Comparing to the base scenario and keeping the response for Q40 fixed, when we change from path 2 to 4 (complex limiting jury instruction introduced), the odds ratio of awarding liability is 0.5524, Interm of percentage its 88% (7% increase)compare to path 2.

Interpretation of Liability from Path 3,7 vs 4,8:

When we change from path 3 (simple limiting jury instruction) to 4 (complex limiting jury instruction) , the odds ratio of awarding liability reduces by 1%.

Question 2:

Question:

 With respect to the 2nd questions, again answers from participants in versions 1,2 and 5 and 6 are meaningless. They did not receive the jury instruction telling them to ignore the evidence. Again, we should do the same analysis as above. Do people that say they can ignore the evidence have lower liability verdicts than people that say they cannot ignore the evidence (for the remaining scenarios 3-4 and 7-8).

Solution:

The question 41 is:

'Were you able to ignore the fact that X5 added core inserts to the later Carve 3000 model when deciding whether the original Carve 3000 was defective?'

The responses for this question can be :

- Yes, I was able to ignore that evidence (1)
- No, I was not able to ignore that evidence (3)

At first we removed the observations with value (1,3). And we have already replaced the path 7,8 with path 3 and 4

In [25]:

```
newdf_3_4 = data14[data14['Path'].isin([3,4]) & data14.Q41.isin(['1','3'])]
```

Now let build a table with **Total liability count** for both **path** and for **both response to Question 41**.

In [26]:

```
a1 = newdf_3_4['Q41'].replace(['1','3'], ['Yes','No'])
a1 = a1[a1.apply(len) > 0]

b1 = data14['Liability']

c =  newdf_3_4.Path
c = c.astype(str)
c.replace(['3','4'],['3 & 7','4 & 8'],inplace = True)

p = pd.crosstab([a1,b1], c,  margins=True , margins_name='Total')
p
```

Out[26]:

	Path	3 & 7	4 & 8	Total
Q41	**Liability**			
No	**No**	8	7	15
	Yes	29	19	48
Yes	**No**	85	86	171
	Yes	42	39	81
Total		164	151	315

Observation (Verdict rate for each path)

For Path (3,7)

> When the says that **he can ignore the fact that X5 added core inserts to the later Carve 3000 model i:e** Yes **to Q41, the verdict rate is** 38%.**
>
> When the juror says that **he can not ignore the fact that X5 added core inserts to the later Carve 3000 model i:e** No **to Q41, the verdict rate increases significantly to** 86%.**

For Path (4,8)

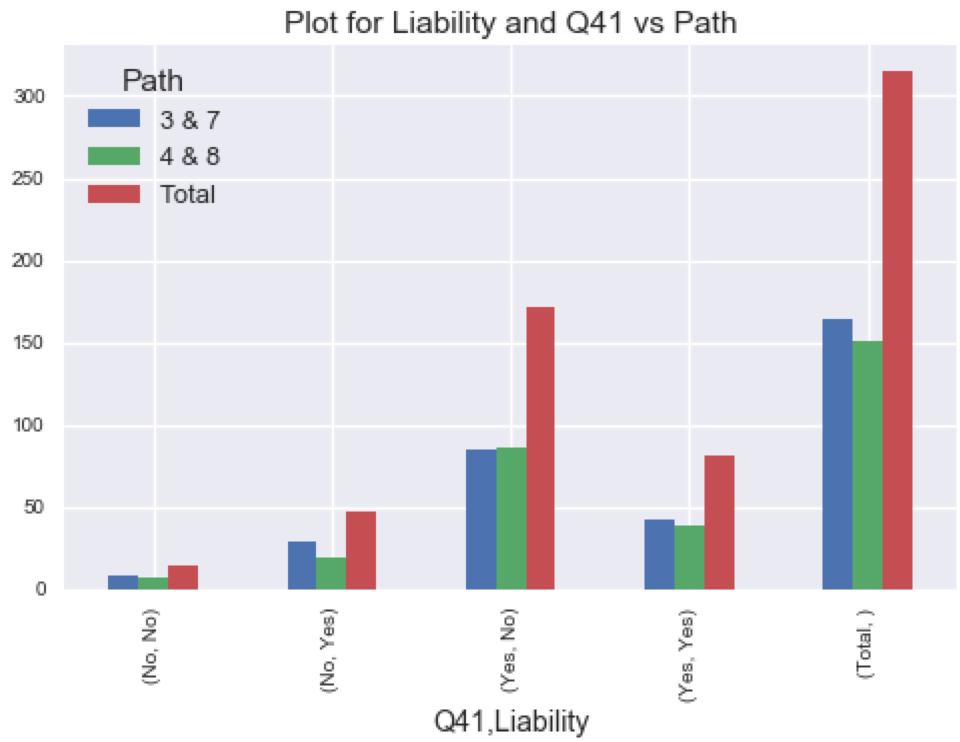
> When the juror says that **he can ignore the fact that X5 added core inserts to the later Carve 3000 model i:e** Yes **to Q41, the verdict rate is** 34%.**
>
> When the juror says that **he can not ignore the fact that X5 added core inserts to the later Carve 3000 model i:e** No **to Q41, the verdict rate increases significantly to** 92%.**

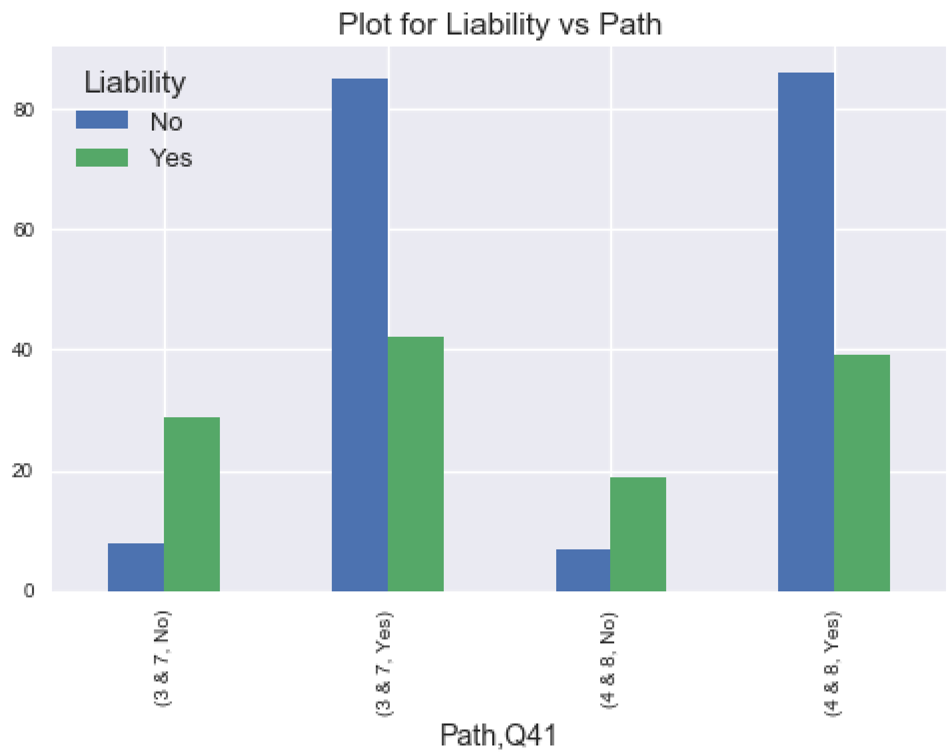
Overal Winrate for **path (3 & 7) is 48% and for path (4 & 8) is 44%.**

Plot for Path (3,7) and (4,8)

In [27]:

```
_ = p.plot(kind='bar', fontsize = 10, figsize=(8,5),
          title='Plot for Liability and Q41 vs Path')
_ = pd.crosstab([c, a1],b1).plot(kind='bar', fontsize = 10, figsize=(8,5),
                                 title='Plot for Liability vs Path')
```





In [28]:

```
import statsmodels.formula.api as smf # stats model formula
import seaborn as sb # statistical visulaization
%matplotlib inline
import matplotlib.pyplot as plt
sb.set(style="darkgrid", context="talk")
from scipy import stats
import statsmodels.formula.api as sm

newdf_3_4['Liability'] = newdf_3_4['Liability'].map({"Yes":1, "No":0})
newdf_3_4.Q41 =  newdf_3_4.Q41.astype('category')
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: Sett
ingWithCopyWarning:
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/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:3643:
SettingWithCopyWarning:
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/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

In [29]:

```
results = smf.logit(formula= 'Liability ~ Q41', data = newdf_3_4).fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.612128
         Iterations 5
```

Out[29]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	315
Model:	Logit	**Df Residuals:**	313
Method:	MLE	**Df Model:**	1
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.09540
Time:	17:44:02	**Log-Likelihood:**	-192.82
converged:	True	**LL-Null:**	-213.16
		LLR p-value:	1.802e-10

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.7472	0.135	-5.540	0.000	-1.012	-0.483
Q41[T.3]	1.9104	0.325	5.876	0.000	1.273	2.548

In [30]:

```
print(np.exp(-0.5705) /(1+np.exp(-0.5705)))
print(np.exp(2.6500-0.5705) /(1+np.exp(2.6500-0.5705)))
```

```
0.3611214604671143
0.8888946624188838
```

Model Observations :

From the Model, we can see that the response to Q41 is a significant factor in deciding Liability. As the p-val is very less (<0.05).

Now

> The odds of finding liability when juror says that he can ignore the fact that X5 added core inserts to the later Carve 3000 model i:e **Yes** to Q41 is : -0.5705 interm of percentage its **36.11%**
>
> The odds ratio of finding liability when juror says that he can not ignore the fact that X5 added core inserts to the later Carve 3000 model i:e **No** to Q41 is :2.6500 and interm of percentage its **89%**

** We find that Q41 is a significant factor for awarding liability**

Logistic regression with categorical variables Path(1-4) and Income

- **Impact of Path and Income on Liability**

$$Liability = \beta_0 + \beta_1 Path$$

In [31]:

```
import statsmodels.formula.api as smf # stats model formula
import seaborn as sb # statistical visulaization
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
sb.set(style="darkgrid", context="talk")
```

In [32]:

```
import pandas as pd
model1_data = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
model1_data['Path'].replace([5,6,7,8], [1,2,3,4], inplace = True)
model1_data['Liability'] = model1_data['Liability'].map({"Yes":1, "No":0})
model1_data['Path']=model1_data['Path'].astype('category')
model1_data['Income']=pd.Categorical(model1_data['Income'])
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

logit_model = smf.logit(formula= 'Liability~ Path+Income', data = model1_data).f
it()
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.666252
         Iterations 4
```

Out[32]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	729
Model:	Logit	**Df Residuals:**	720
Method:	MLE	**Df Model:**	8
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.01197
Time:	17:44:02	**Log-Likelihood:**	-485.70
converged:	True	**LL-Null:**	-491.58
		LLR p-value:	0.1618

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.7719	0.419	-1.841	0.066	-1.594	0.050
Path[T.2]	0.5152	0.220	2.341	0.019	0.084	0.947
Path[T.3]	0.4970	0.220	2.259	0.024	0.066	0.928
Path[T.4]	0.2822	0.225	1.256	0.209	-0.158	0.723
Income[T.2]	0.2845	0.435	0.653	0.514	-0.569	1.138
Income[T.3]	0.0042	0.430	0.010	0.992	-0.839	0.847
Income[T.4]	-0.1094	0.423	-0.259	0.796	-0.938	0.719
Income[T.5]	0.1254	0.456	0.275	0.783	-0.767	1.018
Income[T.6]	0.7400	0.787	0.940	0.347	-0.803	2.283

- Cross tab for Liability vs Path and Liability vs Income

In [33]:

```
c =  model1_data.Liability
c = c.astype(str)
c.replace(['1','0'],['Yes','No'],inplace = True)
pd.crosstab(c,model1_data.Path,margins=True)
```

Out[33]:

Path	1	2	3	4	All
Liability					
No	118	104	106	107	435
Yes	58	84	84	68	294
All	176	188	190	175	729

In [34]:

```
pd.crosstab(c,model1_data.Income,margins=True)
```

Out[34]:

Income	1	2	3	4	5	6	All
Liability							
No	17	80	111	170	53	4	435
Yes	10	70	71	101	37	5	294
All	27	150	182	271	90	9	729

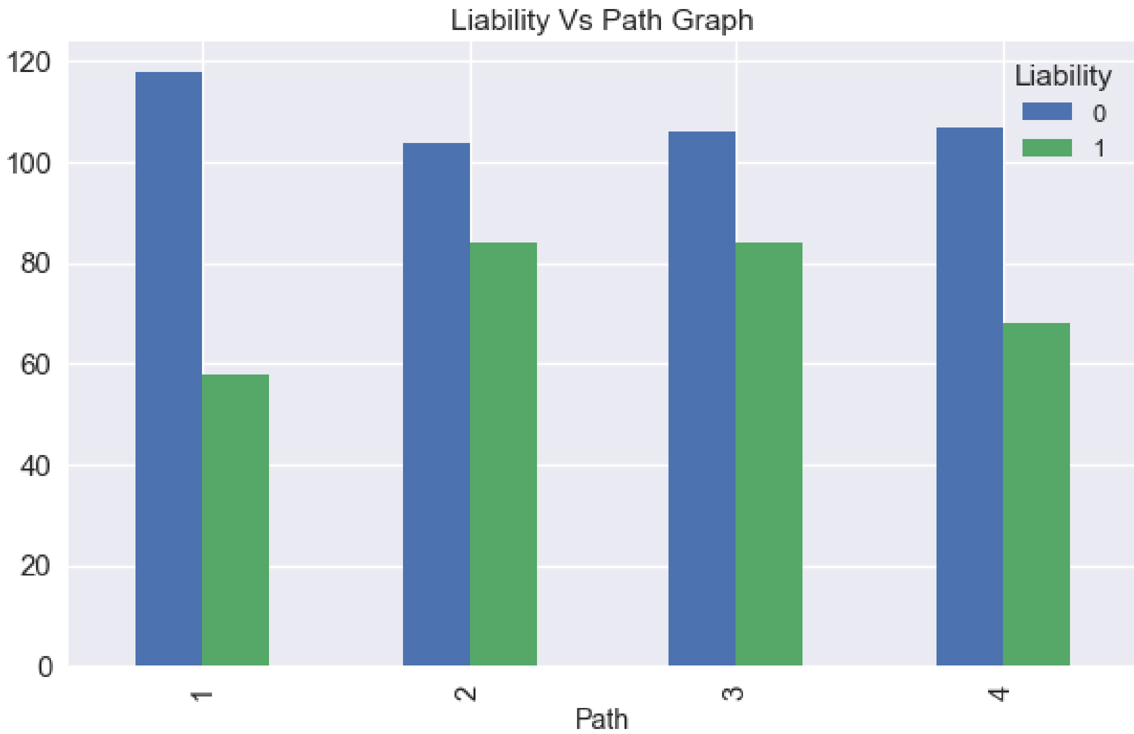
Graph of Liability vs Path

In [35]:

```
pd.crosstab(model1_data.Path,model1_data.Liability).plot(kind='bar', fontsize =
15, figsize=(10,6))
plt.title('Liability Vs Path Graph')
```

Out[35]:

```
Text(0.5,1,'Liability Vs Path Graph')
```



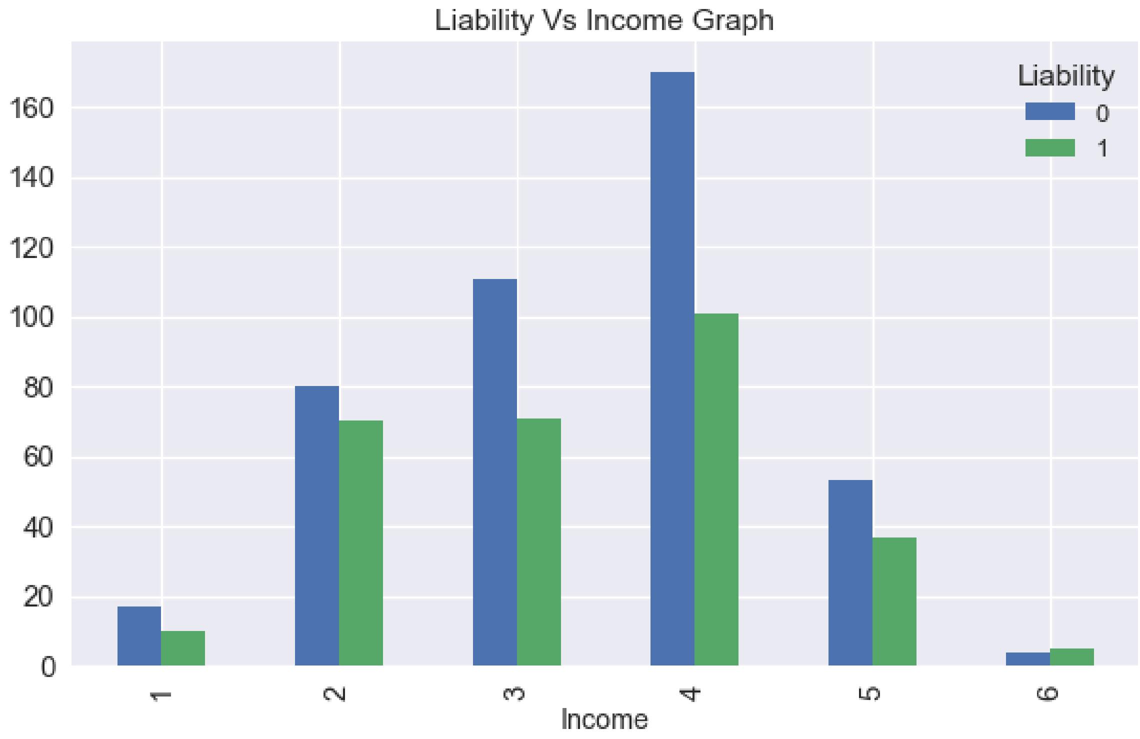
Graph of Liability vs Income

In [36]:

```
pd.crosstab(model1_data.Income,model1_data.Liability).plot(kind='bar', fontsize
= 15, figsize=(10,6))
plt.title('Liability Vs Income Graph')
```

Out[36]:

```
Text(0.5,1,'Liability Vs Income Graph')
```



Interpretation:

From the model(p>0.05) and graphs we can see that Path and Income are not significant in predicting the Liability

Logistic regression for Liability vs Path(1-4) and Path(5-8)

- Impact of Low Anchor and No Anchor on Liability
- $Liability = \beta_0 + \beta_1 Path$

For Performing this regression, we have grouped path1 to path4 into one group and path5 to path8 into other group.

In [37]:

```
df_model=pd.DataFrame(data18[["Liability",'Path']])

df_model['Path']= df_model.Path.astype('int')
df_model['Liability'] = df_model['Liability'].map({"Yes":1, "No":0})
df_model['Path'].replace([1,2,3,4], [14,14,14,14], inplace = True)
df_model['Path'].replace([5,6,7,8], [58,58,58,58], inplace = True)
```

- Running the logit model on Liability vs Path

In [38]:

```
import statsmodels.formula.api as smf # stats model formula
import seaborn as sb # statistical visulaization
%matplotlib inline
import matplotlib.pyplot as plt
sb.set(style="darkgrid", context="talk")

from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

logit_model = smf.logit(formula= 'Liability~C(Path)', data = df_model).fit()
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.673722
         Iterations 4
```

Out[38]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	729
Model:	Logit	**Df Residuals:**	727
Method:	MLE	**Df Model:**	1
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.0008921
Time:	17:44:03	**Log-Likelihood:**	-491.14
converged:	True	**LL-Null:**	-491.58
		LLR p-value:	0.3490

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-0.3214	0.106	-3.027	0.002	-0.530	-0.113
C(Path)[T.58]	-0.1415	0.151	-0.936	0.349	-0.438	0.155

crosstab of Liability and Path(path1-4 as one group and path 5-8 into other group)

In [39]:

```
c =  df_model.Liability
c = c.astype(str)
c.replace(['1','0'],['Yes','No'],inplace = True)
pd.crosstab(c,df_model.Path,margins=True)
```

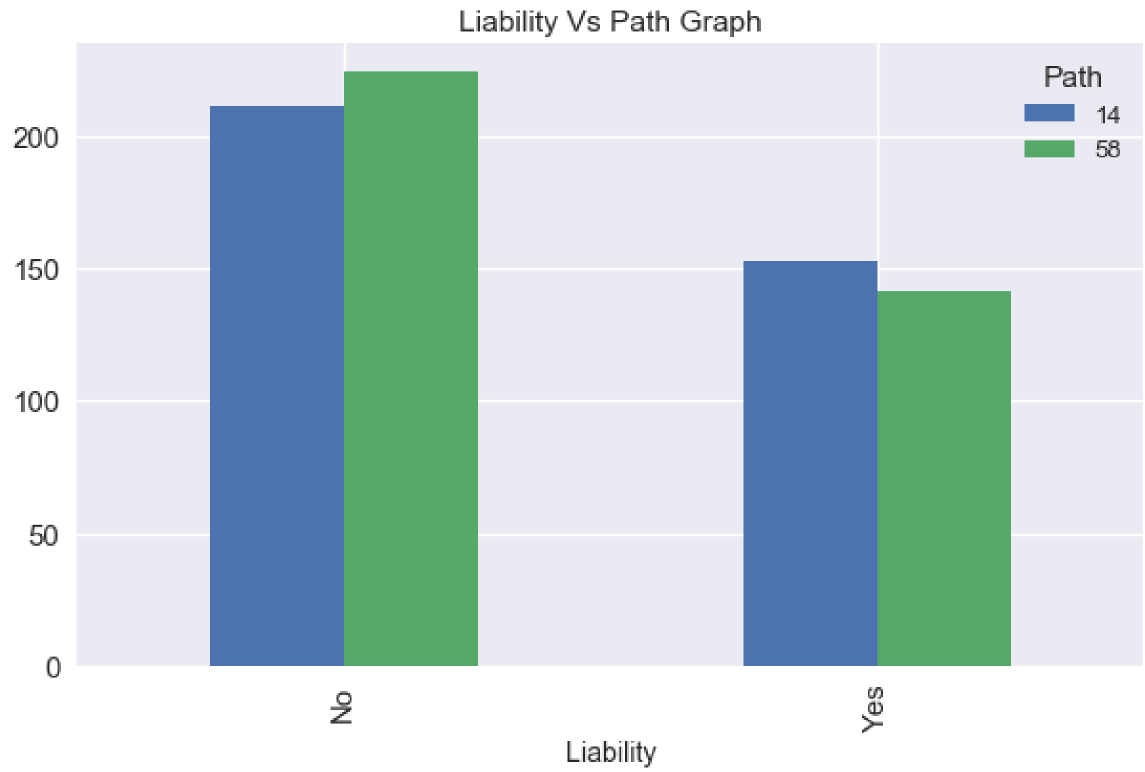
Out[39]:

Path	14	58	All
Liability			
No	211	224	435
Yes	153	141	294
All	364	365	729

Graph of Liability vs Path

In [40]:

```
pd.crosstab(c,df_model.Path).plot(kind='bar', fontsize = 15, figsize=(10,6))
plt.title('Liability Vs Path Graph')

plt.savefig('Juror Response vs Path')
```



In [41]:

```
print(1-np.exp(-0.0382))
```

0.0374795824440034

Interpretation:

- p-value is greater than 0.05.Hence model is not significant.
- we can interpret model coefficient as Compared to path 1-4, there is 3.74% reduction odd in saying yes for path 5-8.

Logistic Regression from Path (1-4) vs Liability for Snowboard and Staircase Dataset.

To perform the regression , we have merged both the dataset.

> For staircase dataset we are taking path (2,3,4 and 5).
>
> **Path(2,3,4,5) from Staircase + (1,2,3,4) no low anchor data from Snowboard + (5,6,7,8) low anchor data from Snowboard**

In [42]:

```
import pandas as pd

staircase_data = pd.read_csv('old_data.csv', encoding= 'ISO-8859-1')
```

In [43]:

```
staircase_data.rename(columns={"Scenario": "Path"},inplace=True)
staircase_data = staircase_data.query("Path>1")
staircase_data.Path.replace([2,3,4,5],[1,2,3,4],inplace=True)

staircase_dat = staircase_data[["Path","Liability","Income"]]
snowboard_dat = data14[["Path","Liability","Income"]]
```

In [44]:

```
snowboard_dat.Income.unique()
snowboard_dat.Income.dtype
```

Out[44]:

```
dtype('int64')
```

In [45]:

```
staircase_dat.Income.unique()
staircase_dat.Income.dtype
```

Out[45]:

```
dtype('int64')
```

Lets see the count of each data set.

In [46]:

```
print(staircase_dat.count())
print(snowboard_dat.count())
```

```
Path         778
Liability    778
Income       778
dtype: int64
Path         729
Liability    729
Income       729
dtype: int64
```

Lets merge the data.

In [47]:

```
merged_data =[staircase_dat,snowboard_dat]
result = pd.concat(merged_data)
```

Lets check if any value is Null or not.

In [48]:

```
result.isnull().sum()
```

Out[48]:

```
Path         0
Liability    0
Income       0
dtype: int64
```

In [49]:

```
result.Liability = result.Liability.astype(str)
result.Liability = result.Liability.replace(['Yes','No'], ['1','0'])
result.Liability = result.Liability.astype(int)

pd.crosstab(result.Path,result.Liability, margins = True)
```

Out[49]:

Liability	**0**	**1**	**All**
Path			
1	219	154	373
2	162	212	374
3	187	207	394
4	188	178	366
All	756	751	1507

In [50]:

```
from scipy import stats
final_model= smf.logit(formula= 'Liability ~ C(Path)', data = result,).fit()
final_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.686718
         Iterations 4
```

Out[50]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	1507
Model:	Logit	**Df Residuals:**	1503
Method:	MLE	**Df Model:**	3
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.009268
Time:	17:44:04	**Log-Likelihood:**	-1034.9
converged:	True	**LL-Null:**	-1044.6
		LLR p-value:	0.0002301

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.3521	0.105	-3.348	0.001	-0.558	-0.146
C(Path)[T.2]	0.6211	0.148	4.192	0.000	0.331	0.911
C(Path)[T.3]	0.4537	0.146	3.113	0.002	0.168	0.739
C(Path)[T.4]	0.2975	0.148	2.006	0.045	0.007	0.588

Interpretations:

> The P-value of Path 2 and 3 are really **very low <0.05**. so we can say that this factor is a significant factor for awarding liability.
>
> The Percentage of awarding Liability at

- Path 1 ---- 45%,
- Path 2 ---- 59%,
- Path 3 ---- 55%,
- Path 4 ---- 51%

> So we can say that there is an **increase in awarding Liability when the remedial measures introduced in Path 2**. But when the **limiting jury instruction introduces the Liability decreases to 55%** and again **it decreases, even more, when explaining to the limiting jury instruction introduced in Path 4.**

Logistic Regression from Path (1-4) Liability vs Income for Snowboard and Staircase Dataset.

To perform the regression , we have merged both the dataset.

> For staircase dataset we are taking path (2,3,4 and 5).
>
> **Path(2,3,4,5) from Staircase + (1,2,3,4) no low anchor data from Snowboard + (5,6,7,8) low anchor data from Snowboard**

In [51]:

```
pd.crosstab(result.Income,result.Liability, margins = True)
```

Out[51]:

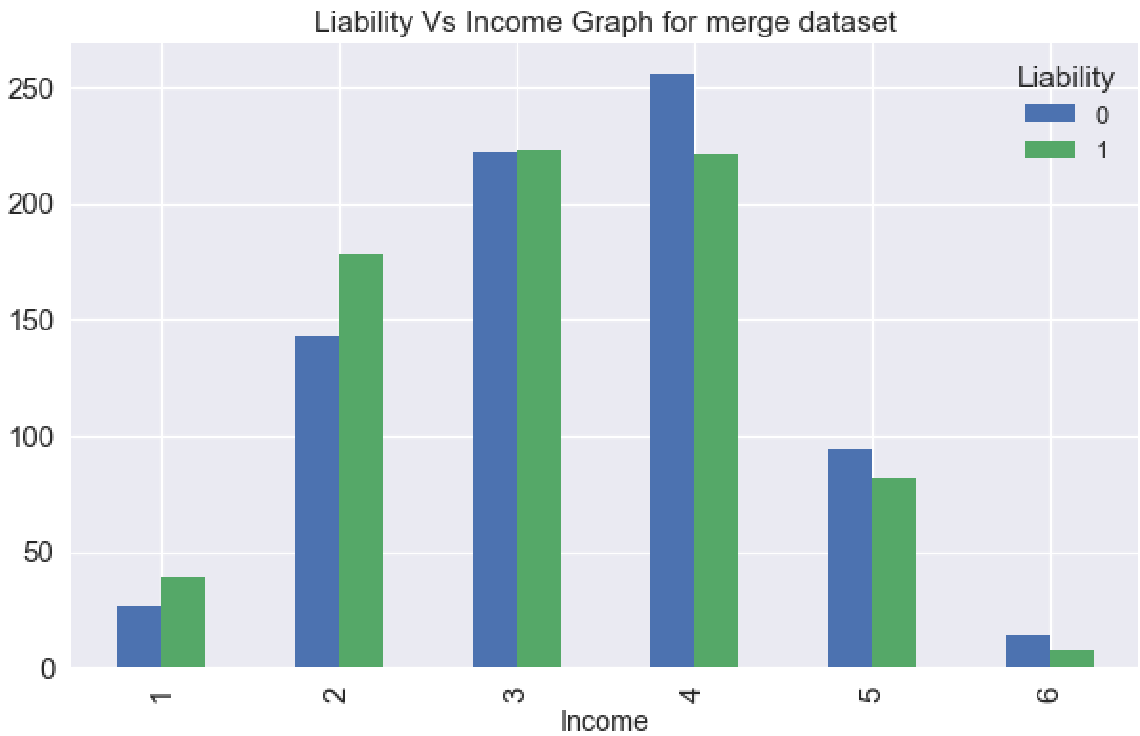
Liability	0	1	All
Income			
1	27	39	66
2	143	178	321
3	222	223	445
4	256	221	477
5	94	82	176
6	14	8	22
All	756	751	1507

In [52]:

```
pd.crosstab(result.Income,result.Liability).plot(kind='bar', fontsize = 15, figs
ize=(10,6))
plt.title('Liability Vs Income Graph for merge dataset')
```

Out[52]:

```
Text(0.5,1,'Liability Vs Income Graph for merge dataset')
```



In [53]:

```
from scipy import stats
final_model= smf.logit(formula= 'Liability ~ C(Income)', data = result).fit()
final_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.689475
         Iterations 4
```

Out[53]:

Logit Regression Results

Dep. Variable:	Liability	No. Observations:	1507
Model:	Logit	Df Residuals:	1501
Method:	MLE	Df Model:	5
Date:	Thu, 28 Jun 2018	Pseudo R-squ.:	0.005289
Time:	17:44:04	Log-Likelihood:	-1039.0
converged:	True	LL-Null:	-1044.6
		LLR p-value:	0.05040

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	0.3677	0.250	1.469	0.142	-0.123	0.858
C(Income)[T.2]	-0.1488	0.274	-0.542	0.588	-0.687	0.389
C(Income)[T.3]	-0.3632	0.268	-1.357	0.175	-0.888	0.161
C(Income)[T.4]	-0.5147	0.267	-1.930	0.054	-1.037	0.008
C(Income)[T.5]	-0.5043	0.292	-1.725	0.085	-1.077	0.069
C(Income)[T.6]	-0.9273	0.509	-1.822	0.068	-1.925	0.070

Interpretations

From the model(p value > 0.05) and graphs, we can see that Income is not a significant factor in the decision of Juror.

Linear model to see the Discounted_Damages vs Anchoring(Path 1-4 vs path 5-8)

- Impact of Anchoring on Discounted damages

Loading the data of snowboard and checking the unique values of Path

In [54]:

```
Anchor_data = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
Anchor_data.Path.unique()
```

Out[54]:

```
array([5, 1, 2, 7, 6, 3, 4, 8])
```

Grouping 1-4 Paths in one group with name "14" and 5-8 in other group with name "58"

In [55]:

```
Anchor_dat=Anchor_data[['Path','Liability','Discounted_Damages']]
```

In [56]:

```
Anchor_dat['Path'].replace([5,6,7,8], [58,58,58,58], inplace = True)
Anchor_dat['Path'].replace([1,2,3,4], [14,14,14,14], inplace = True)
Anchor_dat.Path.unique()
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:4619:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(new_data)
```

Out[56]:

```
array([58, 14])
```

In [57]:

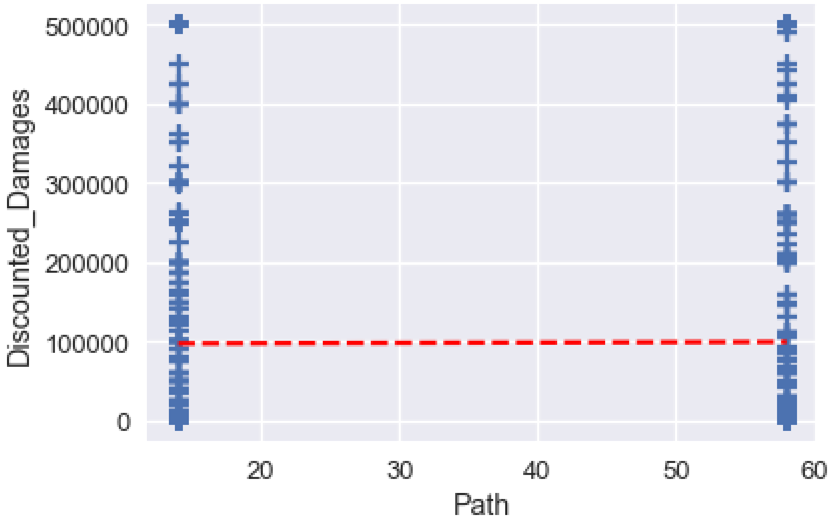
```
Anchor_dat.groupby("Path").Discounted_Damages.mean()
```

Out[57]:

```
Path
14    97380.494505
58    99387.917808
Name: Discounted_Damages, dtype: float64
```

In [58]:

```
plt.scatter(Anchor_dat.Path, Anchor_dat.Discounted_Damages, marker = "+")
plt.plot([14,58], [np.mean(Anchor_dat.query('Path == 14').Discounted_Damages), n
p.mean(Anchor_dat.query('Path == 58').Discounted_Damages)],'r--')
plt.ylabel("Discounted_Damages")
plt.xlabel("Path")
plt.show()
```



Fitting model for Damages vs Path(14 vs 58)

In [59]:

```
Anchor_dat.sample(10)
pd.crosstab(Anchor_dat.Path,Anchor_dat.Liability,margins=True)
```

Out[59]:

Liability	No	Yes	All
Path			
14	211	153	364
58	224	141	365
All	435	294	729

Verifying the non-zero values in each column

In [60]:

```
Anchor_dat.astype(bool).sum(axis=0)
```

Out[60]:

```
Path                  729
Liability             729
Discounted_Damages    294
dtype: int64
```

In [61]:

```
Anchor_dat.sample(10)
```

Out[61]:

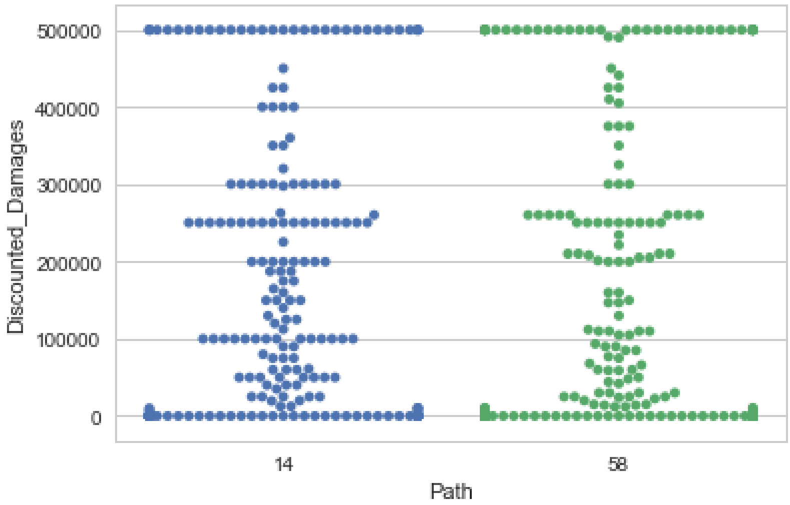
	Path	Liability	Discounted_Damages
470	58	No	0.0
635	58	No	0.0
411	58	No	0.0
190	58	No	0.0
685	14	Yes	250000.0
50	58	Yes	500000.0
500	14	Yes	10000.0
649	58	Yes	85000.0
202	58	Yes	500000.0
594	14	No	0.0

In [62]:

```
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
#sns.boxplot(x="Path", y="Discounted_Damages", data=Anchor_dat)
sns.swarmplot(x="Path", y="Discounted_Damages", data=Anchor_dat)
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10945a470>
```



In [63]:

```
Anchor_dat.Discounted_Damages.describe()
```

Out[63]:

```
count       729.000000
mean      98385.582990
std      165385.417954
min           0.000000
25%           0.000000
50%           0.000000
75%      147000.000000
max      500000.000000
Name: Discounted_Damages, dtype: float64
```

In [64]:

```
# create a fitted model in one line
lm = smf.ols(formula='Discounted_Damages ~ C(Path)', data=Anchor_dat).fit()
# print the coefficients
lm.summary()
```

Out[64]:

OLS Regression Results

Dep. Variable:	Discounted_Damages	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.001
Method:	Least Squares	**F-statistic:**	0.02681
Date:	Thu, 28 Jun 2018	**Prob (F-statistic):**	0.870
Time:	17:44:08	**Log-Likelihood:**	-9793.6
No. Observations:	729	**AIC:**	1.959e+04
Df Residuals:	727	**BIC:**	1.960e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9.738e+04	8674.351	11.226	0.000	8.04e+04	1.14e+05
C(Path)[T.58]	2007.4233	1.23e+04	0.164	0.870	-2.21e+04	2.61e+04

Omnibus:	176.766	**Durbin-Watson:**	2.049
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	316.323
Skew:	1.545	**Prob(JB):**	2.05e-69
Kurtosis:	3.931	**Cond. No.**	2.62

Interpretation

From the plotted graphs and model, we can say there is no significant impact of Discounted damages in low anchor vs no anchor

According to your request, removing the Zero values for discounted damages(Removing Liability=No) and doing regression

In [65]:

```
Anchor_dat1 = Anchor_dat.loc[(Anchor_dat.Discounted_Damages != 0),:]
```

In [66]:

```
Anchor_dat1.sample(10)
```

Out[66]:

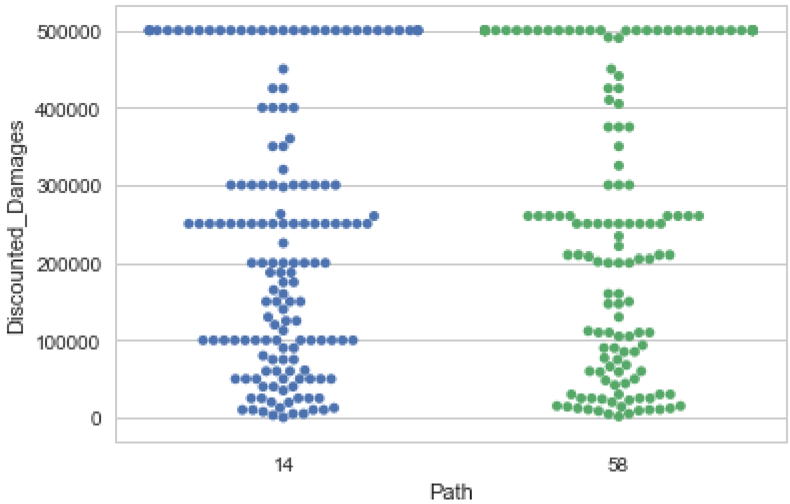
	Path	Liability	Discounted_Damages
109	14	Yes	10000.0
35	14	Yes	360000.0
533	14	Yes	35000.0
457	14	Yes	100000.0
363	14	Yes	100000.0
334	14	Yes	500000.0
516	14	Yes	100000.0
545	14	Yes	250000.0
202	58	Yes	500000.0
693	58	Yes	500000.0

In [67]:

```
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
sns.swarmplot(x="Path", y="Discounted_Damages", data=Anchor_dat1)
```

Out[67]:

<matplotlib.axes._subplots.AxesSubplot at 0x1a150206d8>



In [68]:

```
lm = smf.ols(formula='Discounted_Damages ~ C(Path)', data=Anchor_dat1).fit()
lm.summary()
```

Out[68]:

OLS Regression Results

Dep. Variable:	Discounted_Damages	**R-squared:**	0.005
Model:	OLS	**Adj. R-squared:**	0.002
Method:	Least Squares	**F-statistic:**	1.491
Date:	Thu, 28 Jun 2018	**Prob (F-statistic):**	0.223
Time:	17:44:08	**Log-Likelihood:**	-3973.2
No. Observations:	294	**AIC:**	7950.
Df Residuals:	292	**BIC:**	7958.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.317e+05	1.45e+04	15.952	0.000	2.03e+05	2.6e+05
C(Path)[T.58]	2.56e+04	2.1e+04	1.221	0.223	-1.57e+04	6.69e+04

Omnibus:	643.176	**Durbin-Watson:**	1.907
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	25.812
Skew:	0.252	**Prob(JB):**	2.48e-06
Kurtosis:	1.639	**Cond. No.**	2.57

Interpretations

From the model and graphs we can say there is no significant impact of Discounted damages in low anchor vs no anchor

Linear model to see the Discounted_Damages vs Path

- Impact of Path on Discounted damages

In [69]:

```
import statsmodels.formula.api as smf

# create a fitted model in one line
lm = smf.ols(formula='Discounted_Damages ~ Path', data=data14).fit()

# print the coefficients
lm.summary()
```

Out[69]:

OLS Regression Results

Dep. Variable:	Discounted_Damages	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.6876
Date:	Thu, 28 Jun 2018	**Prob (F-statistic):**	0.407
Time:	17:44:08	**Log-Likelihood:**	-9793.2
No. Observations:	729	**AIC:**	1.959e+04
Df Residuals:	727	**BIC:**	1.960e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	8.686e+04	1.52e+04	5.717	0.000	5.7e+04	1.17e+05
Path	4612.8463	5562.922	0.829	0.407	-6308.462	1.55e+04

Omnibus:	177.256	**Durbin-Watson:**	2.048
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	317.667
Skew:	1.547	**Prob(JB):**	1.05e-69
Kurtosis:	3.942	**Cond. No.**	7.55

Table and Graph of Discounted Damages vs Path

In [70]:

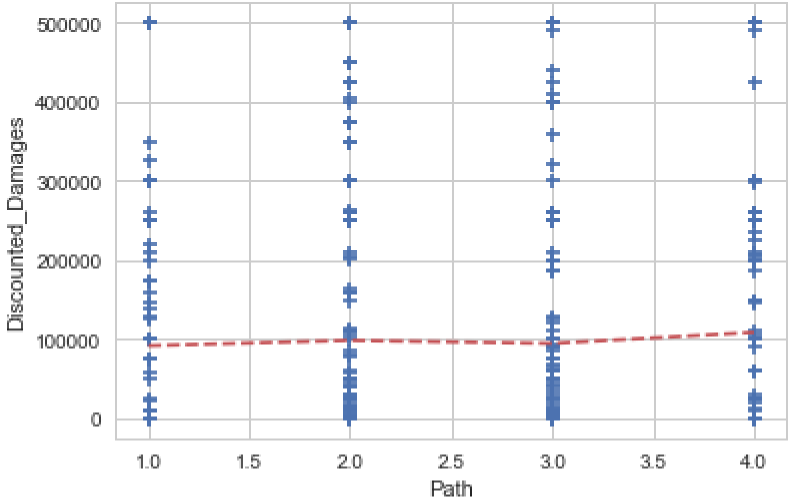
```
data14.groupby("Path").Discounted_Damages.mean()
```

Out[70]:

```
Path
1     91850.795455
2     98567.021277
3     94718.421053
4    108744.285714
Name: Discounted_Damages, dtype: float64
```

In [71]:

```
plt.scatter(data14.Path, data14.Discounted_Damages, marker = "+")
plt.plot([1,2,3,4], [np.mean(data14.query('Path == 1').Discounted_Damages), np.m
ean(data14.query('Path == 2').Discounted_Damages),
                     np.mean(data14.query('Path == 3').Discounted_Damages),np.me
an(data14.query('Path == 4').Discounted_Damages)], 'r--')
plt.ylabel("Discounted_Damages")
plt.xlabel("Path")
plt.show()
```



Interpretation:

- **From the model and the graph we can see that Path is not the predictor for damages.**

Final Regression model(Liability vs Education)

- Impact of Education with levels 1-9(1-Low and 9-High) on Liability

In [72]:

```
import pandas as pd
final_data = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
final_data['Liability'] = final_data['Liability'].map({"Yes":1, "No":0})
final_data['Path'].replace([5,6,7,8], [1,2,3,4], inplace = True)
final_data.Path = pd.Categorical(final_data.Path)
#final_data.Education = pd.Categorical(final_data.Education)
final_data.Income = pd.Categorical(final_data.Income)
```

In [73]:

```
pd.crosstab(final_data.Liability,final_data.Education,margins=True)
```

Out[73]:

Education	1	2	3	4	5	6	7	8	9	All
Liability										
0	1	2	48	101	49	172	55	7	0	435
1	0	2	39	77	35	108	21	7	5	294
All	1	4	87	178	84	280	76	14	5	729

From the above table, we can see that Education=(1,9) are pure classes(probability=1).If we model this we get convergence error.So Removing Education=(1,9) and performing the model

In [74]:

```
final_data=final_data[((final_data.Education != 1) & (final_data.Education != 9
))]
final_data.Education.unique()
```

Out[74]:

```
array([5, 4, 6, 3, 7, 8, 2])
```

In [75]:

```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
final_model= smf.logit(formula= 'Liability ~ C(Education)', data = final_data).f
it()
final_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.667528
         Iterations 5
```

Out[75]:

Logit Regression Results

Dep. Variable:	Liability	**No. Observations:**	723
Model:	Logit	**Df Residuals:**	716
Method:	MLE	**Df Model:**	6
Date:	Thu, 28 Jun 2018	**Pseudo R-squ.:**	0.007982
Time:	17:44:09	**Log-Likelihood:**	-482.62
converged:	True	**LL-Null:**	-486.51
		LLR p-value:	0.2557

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-3.965e-14	1.000	-3.96e-14	1.000	-1.960	1.960
C(Education)[T.3]	-0.2076	1.023	-0.203	0.839	-2.213	1.797
C(Education)[T.4]	-0.2713	1.011	-0.268	0.788	-2.254	1.711
C(Education)[T.5]	-0.3365	1.024	-0.329	0.743	-2.344	1.671
C(Education)[T.6]	-0.4654	1.008	-0.462	0.644	-2.440	1.509
C(Education)[T.7]	-0.9628	1.032	-0.933	0.351	-2.986	1.061
C(Education)[T.8]	3.977e-14	1.134	3.51e-14	1.000	-2.222	2.222

From the model,we can find that Education with levels=(2,3,4,5,6,7,8) has no significant impact on the decision of Juror.But all the participants with Education level=1 agreed that Defendent was negligent and participants with Education level=9 agreed that Defendent was not negligent.

 Merged Dataset Calculations:

We are merging the old dataset(Staircase) and new dataset(Snowboard) and calculating the Case expected and plaintiff win rate.

Replacing the Path of oldset Path=2,3,4,5 to Path = 1,2,3,4 and removing path1

In [76]:

```
import pandas as pd
new_data = pd.read_csv('cleaning.csv', encoding= 'ISO-8859-1')
old_data = pd.read_csv('old_data.csv', encoding= 'ISO-8859-1')
old_data = old_data.query("Scenario>1")
old_data.Scenario.replace([2,3,4,5],[1,2,3,4],inplace=True)
```

Replacing Path of new dataset with Path=5,6,7,8 with Path=1,2,3,4.

In [77]:

```
newdf1=pd.DataFrame(new_data[["StartDate","EndDate","Duration","Liability",'Tota
l_Damages','Path','Was_McNeil_negligent','Discounted_Damages']])
newdf1['Path'].replace([5,6,7,8], [1,2,3,4], inplace = True)
```

In [78]:

```
new_data["Is_substantial"] = np.where(((new_data['Liability'] =='Yes')), 'Yes',
'No')
```

Renaming the column names for old and new datasets

In [79]:

```
newdf1.rename(columns={"StartDate": "Start Date",
                       "EndDate":"End Date",
                       "Total_Damages":"damages",
                       "Was_McNeil_negligent":"Plaintiff_negligent",
                       "Discounted_Damages":"Discounted_damages"
                    },inplace=True)

newdf1['Plaintiff_negligent'] = newdf1['Plaintiff_negligent'].map({1:"Yes", 2:"N
o"})
newdf1['Liability'] = newdf1['Liability'].map({"Yes":1, "No":0})

old_data1=pd.DataFrame(old_data[["Start Date","End Date","Liability",'damages',
'Scenario','Dunn_negligent','Discounted_damages']])
old_data1.rename(columns={
                       "Scenario":"Path","Dunn_negligent":"Plaintiff_negligent"
                         },inplace=True)
```

Merging Old dataset and new dataset.We can retrieve the dataset(old/new) based on the keys(x,y)

In [80]:

```
frames=[newdf1,old_data1]
merge_data = pd.concat(frames, keys=['x', 'y'])
merge_data.head()
merge_data.loc['x'].head()
```

Out[80]:

	Discounted_damages	Duration	End Date	Liability	Path	Plaintiff_negligent	Star Dat
0	0.0	1039.0	2018-04-06 13:32:00	0	1	NaN	2018-04-06 13:15:0
1	0.0	915.0	2018-04-06 13:33:00	0	1	NaN	2018-04-06 13:17:0
2	500000.0	1051.0	2018-04-06 13:33:00	1	1	No	2018-04-06 13:15:0
3	0.0	1092.0	2018-04-06 13:33:00	0	1	NaN	2018-04-06 13:15:0
4	262500.0	1135.0	2018-04-06 13:33:00	1	2	Yes	2018-04-06 13:14:0

In [81]:

```
frames=[newdf1,old_data1]
merge_data = pd.concat(frames, keys=['x', 'y'])
merge_data.head()
```

Out[81]:

		Discounted_damages	Duration	End Date	Liability	Path	Plaintiff_negligent	S D
x	0	0.0	1039.0	2018-04-06 13:32:00	0	1	NaN	2018 04-0(13:15
	1	0.0	915.0	2018-04-06 13:33:00	0	1	NaN	2018 04-0(13:17
	2	500000.0	1051.0	2018-04-06 13:33:00	1	1	No	2018 04-0(13:15
	3	0.0	1092.0	2018-04-06 13:33:00	0	1	NaN	2018 04-0(13:15
	4	262500.0	1135.0	2018-04-06 13:33:00	1	2	Yes	2018 04-0(13:14

In [82]:

```
merge_data.loc['x'].Plaintiff_negligent.unique()
```

Out[82]:

```
array([nan, 'No', 'Yes'], dtype=object)
```

In [83]:

```
#To retrieve data based on the keys:
merge_data.loc['y'].head()
```

Out[83]:

	Discounted_damages	Duration	End Date	Liability	Path	Plaintiff_negligent	S D
101	0.0	NaN	2017-09-29 14:20:00	0	1	NaN	2017 09-2 13:5
102	0.0	NaN	2017-09-29 14:27:00	0	1	NaN	2017 09-2 14:0
103	0.0	NaN	2017-09-29 14:42:00	0	1	NaN	2017 09-2 14:21
104	0.0	NaN	2017-09-29 15:00:00	0	1	NaN	2017 09-2 14:4
105	0.0	NaN	2017-09-29 15:12:00	0	1	NaN	2017 09-2 14:5

Case Expected Value Damages for the merge data Showing the total expected discounted damages mean,median and sd with winrate percentage (entire version)

In [84]:

```
merge_data.query("Path == 1 and Liability == 1").shape
```

Out[84]:

```
(154, 8)
```

In [85]:

```
#merge_data
merge_data['winrate_percentage']=merge_data.Liability
merge_data['Discounted_damages_mean']=merge_data.Discounted_damages
merge_data['Discounted_damages_median']=merge_data.Discounted_damages
merge_data['No.of.Participants']=merge_data.Path
merge_data['Discounted_damages_sd1']=merge_data.Discounted_damages

winrate_damages_expected_merge_data=merge_data.groupby('Path').aggregate(
    {"No.of.Participants":'count','Discounted_damages_sd1': np.std,
    'winrate_percentage': np.mean, 'Discounted_damages_mean': np.mean,'Discounte
d_damages_median':np.median})

winrate_damages_expected_merge_data.winrate_percentage*=100
winrate_damages_expected_merge_data
```

Out[85]:

	No.of.Participants	Discounted_damages_sd1	winrate_percentage	Discounted_
Path				
1	373	148376.777378	41.286863	95551.58176
2	374	143523.832226	56.684492	114567.1122
3	394	134554.225386	52.538071	101647.8426
4	366	140555.065673	48.633880	100403.6885

Contributory negligence is when a juror find the Defendent liable and still thinks that plaintiff is also somehow responsible for the accident

In [86]:

```
#data14.query("Was_McNeil_negligent == 'Yes' & Liability == 'Yes' & Path == 1")
a = merge_data['Path']
a = a.astype(str)
a.replace(['1','2','3','4'],['1 & 5','2 & 6','3 & 7','4 & 8'],inplace = True)
b = merge_data.query("Plaintiff_negligent == 'Yes' & Liability == 1")
c = merge_data.query("Liability == 1")
b = b.rename(columns={'Plaintiff_negligent': 'Contributory_negligence'})
print(pd.crosstab(a,b.Contributory_negligence))
print(pd.crosstab(a,c.Liability))
```

```
Contributory_negligence  Yes
Path
1 & 5                     44
2 & 6                     66
3 & 7                     63
4 & 8                     47
Liability    1
Path
1 & 5      154
2 & 6      212
3 & 7      207
4 & 8      178
```

Finding the Discounted Damages, mean , median and SD when plaintiff wins for the merge data

In [88]:

```
merge_data['Discounted_damages_mean1']=merge_data.Discounted_damages
merge_data['Discounted_damages_median1']=merge_data.Discounted_damages
merge_data['Discounted_damages_sd1']=merge_data.Discounted_damages
winrate_damages_plaintiffwin_merge_data=merge_data.loc[(merge_data['Liability']=
=1)].groupby('Path').aggregate({"No.of.Participants":'count','Discounted_damages
_mean1': np.mean,'Discounted_damages_median1':np.median,'Discounted_damages_sd1'
:np.std})
winrate_damages_plaintiffwin_merge_data
```

Out[88]:

	No.of.Participants	Discounted_damages_mean1	Discounted_damages_media
Path			
1	154	231433.376623	200000.0
2	212	202113.679245	180000.0
3	207	193474.637681	180000.0
4	178	206448.033708	180000.0