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

In [2]:
```
df=pd.read_excel('survey_data3.xlsx')
df.head()
```

Out[2]:

	Person	Age	Gender	State	Children	Salary	Opinion	Agree or Not	Strongly Agree or Not	Neutral or Not	Disagree or Not
0	1	Middle-aged	2	Texas	2	63017	Strongly agree	1	1	0	0
1	2	Middle-aged	2	Virginia	3	100302	Strongly disagree	0	0	0	1
2	3	Middle-aged	2	California	0	144043	Strongly agree	1	1	0	0
3	4	Young	2	California	0	36025	Agree	1	0	0	0
4	5	Middle-aged	1	Texas	0	97543	Neutral	0	0	1	0

In [3]:
```
df=df[['Age','Gender', 'State', 'Children','Salary', 'Neutral or Not']]
df.head()
```

Out[3]:

	Age	Gender	State	Children	Salary	Neutral or Not
0	Middle-aged	2	Texas	2	63017	0
1	Middle-aged	2	Virginia	3	100302	0
2	Middle-aged	2	California	0	144043	0
3	Young	2	California	0	36025	0
4	Middle-aged	1	Texas	0	97543	1

In [4]:
```
df = pd.concat([df,pd.get_dummies(df['Age'], prefix='_', drop_first=True)],axis=1)
df.drop(['Age'],axis=1, inplace=True)
df = pd.concat([df,pd.get_dummies(df['State'], prefix='State', drop_first=True)],axis=1
df.drop(['State'],axis=1, inplace=True)
df['Gender']=df['Gender']-1
df.head()
```

Out[4]:

	Gender	Children	Salary	Neutral or Not	_Middle-aged	_Young	State_California	State_Florida	State_Illinois
0	1	2	63017	0	1	0	0	0	0
1	1	3	100302	0	1	0	0	0	0
2	1	0	144043	0	1	0	1	0	0

	Gender	Children	Salary	Neutral or Not	_Middle-aged	_Young	State_California	State_Florida	State_Illinois
3	1	0	36025	0	0	1	1	0	0
4	0	0	97543	1	1	0	0	0	0

In [5]:
```
df1=df.copy()
```

In [6]:
```
df1['Gend_child']=df1['Gender']*df1['Children']
df1['log_Salary']=np.log(df1['Salary'])
df1['Salary_Sq'] = df1['Salary']**2
df1.head()
```

Out[6]:

	Gender	Children	Salary	Neutral or Not	_Middle-aged	_Young	State_California	State_Florida	State_Illinois
0	1	2	63017	0	1	0	0	0	0
1	1	3	100302	0	1	0	0	0	0
2	1	0	144043	0	1	0	1	0	0
3	1	0	36025	0	0	1	1	0	0
4	0	0	97543	1	1	0	0	0	0

In [7]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [8]:
```
df_copy = df.copy()
train_set = df_copy.sample(frac=0.80, random_state=0)
test_set = df_copy.drop(train_set.index)
train_set.head()
```

Out[8]:

	Gender	Children	Salary	Neutral or Not	_Middle-aged	_Young	State_California	State_Florida	State_Illinoi
132	0	0	97814	0	1	0	0	0	
309	0	2	69817	0	0	0	0	0	
334	0	1	48225	1	0	1	0	0	
196	0	0	37929	0	0	1	0	0	
246	1	2	100062	0	1	0	0	0	

In [9]:
```
test_set.head()
```

Out[9]:

	Gender	Children	Salary	Neutral or Not	_Middle-aged	_Young	State_California	State_Florida	State_Illinois
9	0	3	87457	0	1	0	0	0	1
25	1	2	148075	0	1	0	0	0	0
28	1	2	37963	0	0	1	1	0	0
31	1	2	49505	0	0	1	0	0	0
32	0	3	49723	0	0	1	1	0	0

In [10]:
```
test_set_labels = test_set.pop('Neutral or Not')
train_set_labels = train_set.pop('Neutral or Not')
```

In [11]:
```
model = LogisticRegression(solver = 'liblinear', random_state=0).fit(train_set,train_se
```

In [12]:
```
model.predict_proba(train_set)
```

Out[12]:
```
array([[0.86219691, 0.13780309],
       [0.78731491, 0.21268509],
       [0.71177982, 0.28822018],
       [0.67062899, 0.32937101],
       [0.86712793, 0.13287207],
       [0.8856224 , 0.1143776 ],
       [0.72462848, 0.27537152],
       [0.82804662, 0.17195338],
       [0.75001411, 0.24998589],
       [0.82846528, 0.17153472],
       [0.84788462, 0.15211538],
       [0.8742127 , 0.1257873 ],
       [0.92464217, 0.07535783],
       [0.84759668, 0.15240332],
       [0.89645299, 0.10354701],
       [0.8185525 , 0.1814475 ],
       [0.75454853, 0.24545147],
       [0.88833595, 0.11166405],
       [0.7071119 , 0.2928881 ],
       [0.88322335, 0.11677665],
       [0.80606921, 0.19393079],
       [0.82997053, 0.17002947],
       [0.92140735, 0.07859265],
       [0.69775144, 0.30224856],
       [0.83268065, 0.16731935],
       [0.86432124, 0.13567876],
       [0.87980489, 0.12019511],
       [0.67540205, 0.32459795],
       [0.84765963, 0.15234037],
       [0.88974877, 0.11025123],
       [0.90460006, 0.09539994],
       [0.70408225, 0.29591775],
       [0.886066  , 0.113934  ],
       [0.83998001, 0.16001999],
```

```
[0.70088912, 0.29911088],
[0.87313891, 0.12686109],
[0.93796709, 0.06203291],
[0.70409787, 0.29590213],
[0.6288352 , 0.3711648 ],
[0.8087974 , 0.1912026 ],
[0.69234373, 0.30765627],
[0.8072212 , 0.1927788 ],
[0.8225083 , 0.1774917 ],
[0.86764546, 0.13235454],
[0.79900234, 0.20099766],
[0.91627277, 0.08372723],
[0.66513188, 0.33486812],
[0.75571322, 0.24428678],
[0.7450932 , 0.2549068 ],
[0.8665892 , 0.1334108 ],
[0.92164729, 0.07835271],
[0.75617321, 0.24382679],
[0.67304681, 0.32695319],
[0.95266133, 0.04733867],
[0.85427281, 0.14572719],
[0.75688107, 0.24311893],
[0.86213231, 0.13786769],
[0.71462116, 0.28537884],
[0.86229266, 0.13770734],
[0.80421929, 0.19578071],
[0.73809552, 0.26190448],
[0.82996789, 0.17003211],
[0.8615922 , 0.1384078 ],
[0.69728868, 0.30271132],
[0.8330173 , 0.1669827 ],
[0.77140225, 0.22859775],
[0.89735609, 0.10264391],
[0.78544117, 0.21455883],
[0.89049134, 0.10950866],
[0.59597118, 0.40402882],
[0.87103523, 0.12896477],
[0.77547877, 0.22452123],
[0.9163978 , 0.0836022 ],
[0.78053913, 0.21946087],
[0.93244732, 0.06755268],
[0.70771332, 0.29228668],
[0.77128653, 0.22871347],
[0.80387077, 0.19612923],
[0.82693607, 0.17306393],
[0.66797347, 0.33202653],
[0.81948337, 0.18051663],
[0.85851693, 0.14148307],
[0.74058468, 0.25941532],
[0.79679559, 0.20320441],
[0.83510848, 0.16489152],
[0.73005248, 0.26994752],
[0.77386226, 0.22613774],
[0.84551221, 0.15448779],
[0.75734646, 0.24265354],
[0.86025894, 0.13974106],
[0.85608603, 0.14391397],
[0.82889909, 0.17110091],
[0.79903546, 0.20096454],
[0.82803327, 0.17196673],
[0.88587282, 0.11412718],
[0.80943148, 0.19056852],
[0.84835309, 0.15164691],
[0.70818231, 0.29181769],
[0.79822146, 0.20177854],
```

```
[0.68846905, 0.31153095],
[0.76328708, 0.23671292],
[0.66783209, 0.33216791],
[0.86255512, 0.13744488],
[0.69456338, 0.30543662],
[0.80316342, 0.19683658],
[0.67198988, 0.32801012],
[0.80226091, 0.19773909],
[0.88416168, 0.11583832],
[0.88441871, 0.11558129],
[0.82664881, 0.17335119],
[0.9417015 , 0.0582985 ],
[0.67132427, 0.32867573],
[0.79434738, 0.20565262],
[0.70112094, 0.29887906],
[0.88682458, 0.11317542],
[0.77354716, 0.22645284],
[0.88766289, 0.11233711],
[0.90896645, 0.09103355],
[0.66665834, 0.33334166],
[0.88107375, 0.11892625],
[0.84819385, 0.15180615],
[0.60663992, 0.39336008],
[0.83576054, 0.16423946],
[0.68216649, 0.31783351],
[0.81429324, 0.18570676],
[0.89717638, 0.10282362],
[0.76969533, 0.23030467],
[0.73532111, 0.26467889],
[0.71465557, 0.28534443],
[0.66495231, 0.33504769],
[0.79538054, 0.20461946],
[0.86786273, 0.13213727],
[0.83247944, 0.16752056],
[0.84703402, 0.15296598],
[0.88892592, 0.11107408],
[0.84045316, 0.15954684],
[0.59575449, 0.40424551],
[0.80664293, 0.19335707],
[0.84711173, 0.15288827],
[0.77509012, 0.22490988],
[0.82333329, 0.17666671],
[0.70295222, 0.29704778],
[0.82216595, 0.17783405],
[0.83992709, 0.16007291],
[0.77208252, 0.22791748],
[0.83138907, 0.16861093],
[0.87210749, 0.12789251],
[0.81116332, 0.18883668],
[0.88789261, 0.11210739],
[0.79481859, 0.20518141],
[0.91442957, 0.08557043],
[0.80576133, 0.19423867],
[0.82855318, 0.17144682],
[0.86336652, 0.13663348],
[0.69145657, 0.30854343],
[0.82888579, 0.17111421],
[0.70125056, 0.29874944],
[0.89071964, 0.10928036],
[0.91890224, 0.08109776],
[0.79961881, 0.20038119],
[0.82783298, 0.17216702],
[0.81130972, 0.18869028],
[0.82300584, 0.17699416],
[0.84390398, 0.15609602],
```

```
       [0.72296827, 0.27703173],
       [0.77715848, 0.22284152],
       [0.73957137, 0.26042863],
       [0.81671604, 0.18328396],
       [0.83250036, 0.16749964],
       [0.81877235, 0.18122765],
       [0.91209682, 0.08790318],
       [0.78405102, 0.21594898],
       [0.7869474 , 0.2130526 ],
       [0.79653444, 0.20346556],
       [0.87480725, 0.12519275],
       [0.80419273, 0.19580727],
       [0.93736556, 0.06263444],
       [0.90576013, 0.09423987],
       [0.78160979, 0.21839021],
       [0.84815039, 0.15184961],
       [0.87025829, 0.12974171],
       [0.79187429, 0.20812571],
       [0.7032066 , 0.2967934 ],
       [0.70392209, 0.29607791],
       [0.85417243, 0.14582757],
       [0.79489195, 0.20510805],
       [0.80746904, 0.19253096],
       [0.8318904 , 0.1681096 ],
       [0.72020016, 0.27979984],
       [0.72050982, 0.27949018],
       [0.76728732, 0.23271268],
       [0.7133847 , 0.2866153 ],
       [0.69907816, 0.30092184],
       [0.76489217, 0.23510783],
       [0.93704341, 0.06295659],
       [0.91890363, 0.08109637],
       [0.82058446, 0.17941554],
       [0.87117836, 0.12882164],
       [0.68372926, 0.31627074],
       [0.87437959, 0.12562041],
       [0.76995775, 0.23004225],
       [0.85236049, 0.14763951],
       [0.88735595, 0.11264405],
       [0.84432087, 0.15567913],
       [0.88578751, 0.11421249],
       [0.80421044, 0.19578956],
       [0.86701342, 0.13298658],
       [0.67777711, 0.32222289],
       [0.82894428, 0.17105572],
       [0.78721444, 0.21278556],
       [0.778574  , 0.221426  ],
       [0.85199682, 0.14800318],
       [0.82330875, 0.17669125],
       [0.88164808, 0.11835192],
       [0.7264837 , 0.2735163 ],
       [0.72668481, 0.27331519],
       [0.86028598, 0.13971402],
       [0.8529869 , 0.1470131 ],
       [0.7912464 , 0.2087536 ],
       [0.81784422, 0.18215578],
       [0.71680293, 0.28319707],
       [0.805641  , 0.194359  ],
       [0.8228556 , 0.1771444 ],
       [0.78066755, 0.21933245],
       [0.88742714, 0.11257286],
       [0.71173367, 0.28826633],
       [0.64268202, 0.35731798],
       [0.69384706, 0.30615294],
       [0.83347053, 0.16652947],
```

```
[0.78074778, 0.21925222],
[0.81294008, 0.18705992],
[0.81529467, 0.18470533],
[0.81325061, 0.18674939],
[0.69493709, 0.30506291],
[0.94129468, 0.05870532],
[0.77067086, 0.22932914],
[0.92258031, 0.07741969],
[0.8487891 , 0.1512109 ],
[0.84895504, 0.15104496],
[0.90654606, 0.09345394],
[0.82457597, 0.17542403],
[0.75358553, 0.24641447],
[0.88531632, 0.11468368],
[0.84026454, 0.15973546],
[0.79781354, 0.20218646],
[0.81741374, 0.18258626],
[0.87971367, 0.12028633],
[0.68717291, 0.31282709],
[0.80080563, 0.19919437],
[0.8508538 , 0.1491462 ],
[0.91532594, 0.08467406],
[0.88190604, 0.11809396],
[0.62214286, 0.37785714],
[0.81045303, 0.18954697],
[0.9037245 , 0.0962755 ],
[0.82076103, 0.17923897],
[0.7918712 , 0.2081288 ],
[0.84438985, 0.15561015],
[0.81352093, 0.18647907],
[0.69368775, 0.30631225],
[0.72263774, 0.27736226],
[0.87449485, 0.12550515],
[0.73102671, 0.26897329],
[0.82628585, 0.17371415],
[0.72862776, 0.27137224],
[0.66269743, 0.33730257],
[0.60908864, 0.39091136],
[0.88655713, 0.11344287],
[0.83067043, 0.16932957],
[0.7750607 , 0.2249393 ],
[0.80844057, 0.19155943],
[0.8349742 , 0.1650258 ],
[0.76945931, 0.23054069],
[0.75193877, 0.24806123],
[0.7213132 , 0.2786868 ],
[0.79310126, 0.20689874],
[0.81594027, 0.18405973],
[0.8410029 , 0.1589971 ],
[0.86983226, 0.13016774],
[0.79214912, 0.20785088],
[0.84632342, 0.15367658],
[0.8717055 , 0.1282945 ],
[0.92580037, 0.07419963],
[0.84816729, 0.15183271],
[0.87547094, 0.12452906],
[0.75806575, 0.24193425],
[0.68712858, 0.31287142],
[0.81540757, 0.18459243],
[0.77622207, 0.22377793],
[0.67139459, 0.32860541],
[0.68274337, 0.31725663],
[0.87472305, 0.12527695],
[0.7200188 , 0.2799812 ],
[0.63709234, 0.36290766],
```

```
       [0.69528671, 0.30471329],
       [0.85873084, 0.14126916],
       [0.813299  , 0.186701  ],
       [0.87191919, 0.12808081],
       [0.7626565 , 0.2373435 ],
       [0.76518871, 0.23481129],
       [0.74685518, 0.25314482],
       [0.67189483, 0.32810517],
       [0.83603055, 0.16396945],
       [0.836449  , 0.163551  ],
       [0.86457385, 0.13542615],
       [0.92530953, 0.07469047],
       [0.82828938, 0.17171062],
       [0.85454332, 0.14545668],
       [0.68618879, 0.31381121],
       [0.7173696 , 0.2826304 ],
       [0.8239487 , 0.1760513 ],
       [0.761075  , 0.238925  ],
       [0.80239173, 0.19760827],
       [0.66103604, 0.33896396],
       [0.88154633, 0.11845367],
       [0.85801981, 0.14198019],
       [0.94346767, 0.05653233],
       [0.7309198 , 0.2690802 ],
       [0.69587816, 0.30412184]])
```

In [13]:

```
model.predict(train_set)
```

Out[13]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [14]:

```
model.predict_proba(test_set)
```

Out[14]:

```
array([[0.83746712, 0.16253288],
       [0.94135784, 0.05864216],
       [0.67076976, 0.32923024],
       [0.71667733, 0.28332267],
       [0.71750641, 0.28249359],
       [0.84471226, 0.15528774],
       [0.71063242, 0.28936758],
       [0.85014348, 0.14985652],
       [0.85193535, 0.14806465],
       [0.8236521 , 0.1763479 ],
       [0.91550738, 0.08449262],
       [0.65500372, 0.34499628],
       [0.817187  , 0.182813  ],
       [0.81875845, 0.18124155],
       [0.73453594, 0.26546406],
       [0.67805544, 0.32194456],
```

```
       [0.75449992, 0.24550008],
       [0.7118798 , 0.2881202 ],
       [0.83607423, 0.16392577],
       [0.87015031, 0.12984969],
       [0.93756667, 0.06243333],
       [0.80145971, 0.19854029],
       [0.81506307, 0.18493693],
       [0.91539711, 0.08460289],
       [0.82822271, 0.17177729],
       [0.78670214, 0.21329786],
       [0.73633414, 0.26366586],
       [0.92257228, 0.07742772],
       [0.71538135, 0.28461865],
       [0.9108976 , 0.0891024 ],
       [0.88942101, 0.11057899],
       [0.85001925, 0.14998075],
       [0.87697546, 0.12302454],
       [0.82928425, 0.17071575],
       [0.81531726, 0.18468274],
       [0.79097069, 0.20902931],
       [0.70645534, 0.29354466],
       [0.68635023, 0.31364977],
       [0.81807032, 0.18192968],
       [0.74584729, 0.25415271],
       [0.77365551, 0.22634449],
       [0.76189216, 0.23810784],
       [0.83588659, 0.16411341],
       [0.85755787, 0.14244213],
       [0.69596147, 0.30403853],
       [0.80923766, 0.19076234],
       [0.70741852, 0.29258148],
       [0.73042915, 0.26957085],
       [0.88290782, 0.11709218],
       [0.78646297, 0.21353703],
       [0.66572453, 0.33427547],
       [0.80727079, 0.19272921],
       [0.81212912, 0.18787088],
       [0.69930684, 0.30069316],
       [0.90876304, 0.09123696],
       [0.80671894, 0.19328106],
       [0.82965814, 0.17034186],
       [0.80201098, 0.19798902],
       [0.74978909, 0.25021091],
       [0.8321812 , 0.1678188 ],
       [0.6956877 , 0.3043123 ],
       [0.69436052, 0.30563948],
       [0.81195745, 0.18804255],
       [0.84675205, 0.15324795],
       [0.79070401, 0.20929599],
       [0.89807904, 0.10192096],
       [0.85772495, 0.14227505],
       [0.72610732, 0.27389268],
       [0.85506449, 0.14493551],
       [0.83341849, 0.16658151],
       [0.8138364 , 0.1861636 ],
       [0.92321128, 0.07678872],
       [0.83238269, 0.16761731],
       [0.78317051, 0.21682949],
       [0.82884059, 0.17115941],
       [0.8384776 , 0.1615224 ],
       [0.74123962, 0.25876038],
       [0.83694847, 0.16305153],
       [0.81824603, 0.18175397],
       [0.83078379, 0.16921621]])
```

In [15]:
```
model.predict(test_set)
```

Out[15]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [16]:
```
model.score(train_set,train_set_labels)
```

Out[16]:
```
0.8244514106583072
```

In [17]:
```
model.score(test_set,test_set_labels)
```

Out[17]:
```
0.85
```

In [18]:
```
confusion_matrix(train_set_labels,model.predict(train_set))
```

Out[18]:
```
array([[263,   0],
       [ 56,   0]], dtype=int64)
```

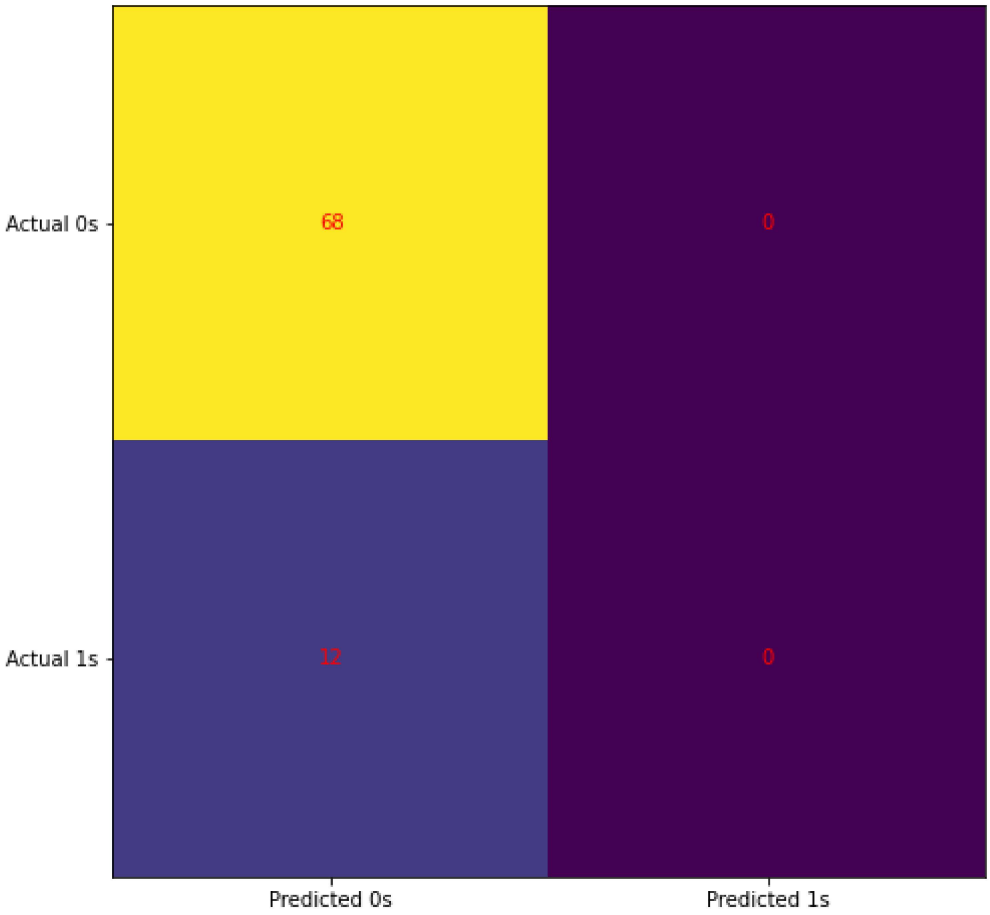
In [19]:
```
confusion_matrix(test_set_labels,model.predict(test_set))
```

Out[19]:
```
array([[68,  0],
       [12,  0]], dtype=int64)
```

In [20]:
```
import matplotlib.pyplot as plt
```

In [21]:
```
cm = confusion_matrix(test_set_labels, model.predict(test_set))

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.show()
```



In [22]:
```
model = LogisticRegression(C=10.0, solver = 'liblinear', random_state=0).fit(train_set,
```

In [23]:
```
model.score(train_set,train_set_labels)
```

Out[23]:
```
0.8244514106583072
```

In [24]:
```
prob1 = model.predict_proba(train_set)
predicted=[0 if i>0.84 else 1 for i in prob1[:,0]]
predicted
```

Out[24]:
```
[0,
 1,
 1,
 1,
 0,
 0,
 1,
 1,
 1,
 1,
 0,
 0,
 0,
 0,
 0,
 1,
 1,
 0,
```

```
1,
0,
1,
1,
0,
1,
1,
0,
0,
1,
0,
0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
1,
1,
1,
1,
0,
1,
0,
1,
1,
1,
0,
0,
1,
1,
0,
0,
1,
0,
1,
0,
1,
1,
1,
0,
1,
1,
1,
0,
1,
0,
1,
0,
1,
0,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
```

```
1,
1,
1,
1,
0,
1,
0,
0,
1,
1,
1,
0,
1,
0,
1,
1,
1,
1,
1,
0,
1,
1,
1,
1,
0,
0,
1,
0,
1,
1,
1,
0,
1,
0,
0,
1,
0,
0,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
0,
1,
0,
0,
0,
1,
1,
0,
1,
1,
1,
1,
1,
1,
1,
0,
1,
```

```
0,
1,
0,
1,
1,
0,
1,
1,
1,
0,
0,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
0,
1,
0,
0,
1,
0,
0,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
1,
1,
1,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
0,
1,
1,
1,
1,
0,
1,
```

```
0,
1,
1,
0,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
1,
1,
1,
0,
1,
0,
0,
0,
0,
1,
1,
0,
0,
1,
1,
0,
1,
1,
0,
0,
0,
1,
1,
0,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
1,
1,
1,
0,
```

```
0,
1,
0,
0,
0,
0,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
0,
1,
0,
1,
1,
1,
1,
1,
1,
1,
0,
0,
1,
0,
1,
1,
1,
1,
1,
1,
0,
0,
0,
1,
1]
```

In [25]:
```
sum(abs(train_set_labels-predicted))
```

Out[25]: 180

In [26]:
```
len(train_set_labels)
```

Out[26]: 319

In [27]:
```
error_percent =sum(abs(train_set_labels-predicted))/319
1-error_percent
```

Out[27]: 0.4357366771159875

In [28]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

In [29]:
```
scaler = StandardScaler()
```

In [30]:
```
scaled_features = scaler.fit_transform(train_set)
```

In [31]:
```
kmeans = KMeans(init="random", n_clusters=5, n_init=10, max_iter=300, random_state=42)
kmeans.fit(scaled_features)
```

Out[31]:
```
KMeans(init='random', n_clusters=5, random_state=42)
```

In [32]:
```
kmeans.inertia_
```

Out[32]:
```
2982.105883164649
```

In [33]:
```
kmeans.cluster_centers_
```

Out[33]:
```
array([[-0.04971743,  0.01781439,  0.33015189,  0.3133736 , -0.52049624,
         0.25232533,  0.26227165, -0.32218974, -0.37864122, -0.29160592,
         0.34644297,  0.32731074, -0.38938997,  0.15103416],
       [ 0.20089178, -0.12825991,  0.34720311,  0.20571739, -0.52049624,
        -0.28522961, -0.33391355, -0.32218974,  1.02486446, -0.29160592,
        -0.33968311, -0.35666298,  1.19350246, -0.31622777],
       [-0.11496944, -0.07072974,  0.03824646,  0.02921187, -0.0321483 ,
        -0.28522961, -0.33391355, -0.32218974, -0.37864122,  3.42928564,
        -0.33968311, -0.35666298, -0.38938997, -0.31622777],
       [-0.29038454,  0.15619483,  0.10758022,  0.24343225, -0.19493094,
        -0.28522961, -0.33391355,  3.10376116, -0.37864122, -0.29160592,
        -0.33968311, -0.35666298, -0.38938997, -0.31622777],
       [ 0.06753546,  0.06443599, -1.23856512, -1.09544512,  1.92124347,
         0.03605657,  0.1174357 , -0.32218974, -0.02037603, -0.29160592,
        -0.06141164,  0.01830333, -0.18888085,  0.33230714]])
```

In [34]:
```
kmeans.n_iter_
```

Out[34]:
```
19
```

In [35]:
```
kmeans.labels_[:]
```

Out[35]:
```
array([0, 2, 4, 4, 0, 1, 4, 0, 2, 1, 0, 0, 1, 1, 2, 0, 0, 0, 4, 0, 0, 1,
       1, 4, 2, 0, 0, 1, 1, 3, 1, 4, 0, 1, 4, 0, 1, 0, 4, 3, 4, 1, 1, 0,
       0, 0, 4, 0, 4, 0, 3, 0, 4, 0, 1, 0, 1, 0, 0, 0, 4, 1, 1, 4, 0, 0,
       0, 0, 2, 0, 1, 0, 0, 0, 0, 4, 1, 1, 2, 1, 3, 0, 2, 1, 0, 4, 1, 0,
       0, 3, 1, 1, 1, 0, 2, 0, 2, 4, 2, 4, 0, 4, 3, 1, 0, 4, 0, 0, 3, 3,
       1, 4, 3, 0, 1, 0, 0, 0, 4, 0, 1, 3, 1, 3, 1, 0, 1, 1, 0, 4, 0, 1,
       3, 0, 0, 0, 4, 1, 3, 0, 1, 4, 1, 1, 2, 1, 0, 0, 0, 4, 1, 0, 1, 0,
       4, 0, 0, 0, 0, 0, 3, 3, 0, 1, 4, 1, 3, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 3, 2, 0, 0, 0, 4, 4, 3, 2, 0, 1, 4, 4, 0, 4, 4, 3, 0, 1, 0, 0,
       4, 3, 1, 2, 1, 0, 0, 1, 1, 4, 1, 4, 0, 1, 0, 0, 4, 4, 3, 0, 0, 0,
       4, 2, 0, 0, 0, 4, 2, 0, 0, 1, 3, 1, 0, 2, 0, 1, 0, 3, 0, 0, 0, 0,
       1, 0, 3, 0, 0, 4, 3, 0, 1, 2, 4, 0, 0, 3, 0, 1, 2, 4, 4, 3, 3, 1,
       4, 4, 4, 0, 0, 4, 0, 0, 0, 1, 4, 3, 1, 0, 1, 0, 0, 0, 2, 0, 0, 2,
       4, 0, 2, 4, 4, 0, 4, 2, 4, 1, 1, 0, 0, 1, 0, 3, 2, 0, 0, 2, 1, 0,
       4, 4, 1, 0, 0, 4, 0, 1, 1, 4, 0])
```

In [36]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [37]:
```
model = KNeighborsClassifier(n_neighbors=1)
```

In [38]:
```
model.fit(scaled_features,train_set_labels)
predicted = model.predict(scaled_features)
predicted
```

Out[38]:
```
array([0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0], dtype=int64)
```

In [39]:
```
sum(abs(predicted-train_set_labels))
```

Out[39]:
```
0
```

In [40]:
```
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
```

In [41]:
```
model = RandomForestClassifier()
model.fit(train_set, train_set_labels)
```

Out[41]:
```
RandomForestClassifier()
```

In [42]:
```
yhat = model.predict(train_set)
```

In [44]:
```
sum(abs(yhat-train_set_labels))
```

Out[44]:
```
0
```

In [53]:
```
df1.describe().transpose()
```

Out[53]:

	count	mean	std	min	25%	50%	
Gender	399.0	5.864662e-01	4.930851e-01	0.000000e+00	0.000000e+00	1.000000e+00	1.0000
Children	399.0	1.313283e+00	1.022166e+00	0.000000e+00	0.000000e+00	2.000000e+00	2.0000

	count	mean	std	min	25%	50%	
Salary	399.0	7.894247e+04	2.723687e+04	2.068700e+04	5.812850e+04	7.921200e+04	9.5796
Neutral or Not	399.0	1.704261e-01	3.764788e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
_Middle-aged	399.0	5.463659e-01	4.984706e-01	0.000000e+00	0.000000e+00	1.000000e+00	1.0000
_Young	399.0	2.180451e-01	4.134366e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_California	399.0	8.270677e-02	2.757843e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Florida	399.0	1.027569e-01	3.040223e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Illinois	399.0	9.523810e-02	2.939121e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Michigan	399.0	1.152882e-01	3.197704e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Minnesota	399.0	7.769424e-02	2.680259e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_New York	399.0	1.152882e-01	3.197704e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Ohio	399.0	1.102757e-01	3.136263e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Texas	399.0	1.203008e-01	3.257213e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
State_Virginia	399.0	9.774436e-02	2.973415e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0000
Gend_child	399.0	8.020050e-01	1.009202e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.0000
log_Salary	399.0	1.121107e+01	3.760868e-01	9.937261e+00	1.097039e+01	1.127988e+01	1.1469
Salary_Sq	399.0	6.971901e+09	4.626720e+09	4.279520e+08	3.379095e+09	6.274541e+09	9.1769

In [46]:
```
from sklearn.neural_network import MLPClassifier
```

In [48]:
```
mlp = MLPClassifier(hidden_layer_sizes=(13,3,2),max_iter=1000)
mlp.fit(scaled_features,train_set_labels)
```

```
C:\Users\Top\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:614: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and t
he optimization hasn't converged yet.
  warnings.warn(
```

Out[48]:
```
MLPClassifier(hidden_layer_sizes=(13, 3, 2), max_iter=1000)
```

In [49]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(17,13,13,13), learning_rate='constant',
       learning_rate_init=0.001, max_iter=1000, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

Out[49]:
```
MLPClassifier(hidden_layer_sizes=(17, 13, 13, 13), max_iter=1000)
```

In [50]:
```
predictions = mlp.predict(train_set) #to predict with the test set you'd also have to s
```

```
predictions
```

Out[50]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [51]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(train_set_labels,predictions)) #this is the same result we got w
#experiment with settings to see if you can improve it
```

```
[[263   0]
 [ 56   0]]
```

In [52]:

```
print(classification_report(train_set_labels,predictions))
```

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       263
           1       0.00      0.00      0.00        56

    accuracy                           0.82       319
   macro avg       0.41      0.50      0.45       319
weighted avg       0.68      0.82      0.75       319

C:\Users\Top\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels w
ith no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Top\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels w
ith no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Top\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels w
ith no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [53]:

```
import re

txt = "The rain in Spain is mainly on the plain."
x = re.findall("ai", txt)
print(x)
```

```
['ai', 'ai', 'ai', 'ai']
```

In [54]:

```
x = re.split("\s", txt)
print(x)
```

```
['The', 'rain', 'in', 'Spain', 'is', 'mainly', 'on', 'the', 'plain.']
```

In [55]:
```
x = re.split("\s", txt, 1) #splits at only the first space
print(x)
```

```
['The', 'rain in Spain is mainly on the plain.']
```

In [56]:
```
x = re.sub("\s", "-", txt)
print(x)
```

```
The-rain-in-Spain-is-mainly-on-the-plain.
```

In [57]:
```
x = re.search(r"\bS\w+", txt) #looks for a word with uppercase S at the start, gives st
print(x.span())
```

```
(12, 17)
```

In [58]:
```
print(x.string) #prints whole string where the word appeared
```

```
The rain in Spain is mainly on the plain.
```

In [59]:
```
print(x.group()) #prints just the word
```

```
Spain
```

In [60]:
```
x = re.findall("[mat]", txt)
print(x)
```

```
['a', 'a', 'm', 'a', 't', 'a']
```

In [61]:
```
x = re.findall("ain+", txt)
print(x)
```

```
['ain', 'ain', 'ain', 'ain']
```

In [ ]:
```
#https://www.w3schools.com/python/python_regex.asp more code keys here
```