project-2

February 21, 2024

[ ]:

```
LOGISTIC REGRESSION BASED ON  User_Data
```

[ ]:

```
Salaries analysis based on logistic regression
```

[73]:

```
import pandas as pd
```

[74]:

```
data=pd.read_csv(r"C:\Users\micro\Downloads\User_Data.csv")
```

[67]:

```
data
```

[67]:

```
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        …       …   …              …        …
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
```

[116]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

[96]:

```
x=data[['EstimatedSalary']]
y=data['Age']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪4,random_state=100)
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model
```

```
[96]: LogisticRegression()
```

```
[100]: model.fit(x_train, y_train)
```

```
C:\Users\micro\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
[100]: LogisticRegression()
```

```
[98]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
       ↪2,random_state=101)
```

```
[99]: y_pred=model.predict(x_test)
      y_pred
```

```
[99]: array([35, 35, 37, 35, 35, 35, 35, 35, 35, 35, 37, 35, 37, 35, 35, 35, 37,
             35, 35, 35, 37, 37, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
             37, 35, 37, 35, 35, 35, 35, 35, 35, 35, 37, 35, 35, 35, 35, 35, 35,
             37, 35, 35, 35, 35, 37, 35, 35, 35, 35, 37, 35, 37, 35, 35, 35, 35,
             35, 35, 35, 35, 37, 35, 35, 35, 35, 35, 35, 35], dtype=int64)
```

```
[101]: y_test
```

```
[101]: 38     26
       387    39
       270    43
       181    31
       195    34
              ..
       130    31
       13     32
       141    18
       304    40
       167    35
       Name: Age, Length: 80, dtype: int64
```

```
[112]: clf=LogisticRegression()
       clf.fit(x_train,y_train)
```

```
C:\Users\micro\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
[112]: LogisticRegression()
```

```
[113]: y_pred=clf.predict(x_test)
       y_pred
```

```
[113]: array([35, 35, 36, 35, 35, 35, 35, 35, 35, 35, 36, 37, 37, 35, 35, 35, 36,
              35, 35, 35, 36, 36, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
              32, 37, 36, 35, 35, 35, 35, 35, 35, 35, 36, 35, 35, 35, 35, 35, 35,
              37, 35, 35, 35, 35, 36, 35, 35, 35, 35, 37, 35, 37, 35, 35, 35, 35,
              35, 35, 35, 35, 36, 35, 35, 35, 35, 35, 35, 35], dtype=int64)
```

```
[102]: from sklearn.metrics import accuracy_score
```

```
[103]: import numpy as np
```

```
[110]: accuracy=accuracy_score(y_test,np.round(y_pred))
```

```
[111]: accuracy
```

```
[111]: 0.1
```

```
[117]: inputdata=[[17]]
       prediction=model.predict(inputdata)
       prediction
```

```
C:\Users\micro\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid feature names,
but LogisticRegression was fitted with feature names
  warnings.warn(
```

```
[117]: array([35], dtype=int64)
```

```
 [47]: data.head()
```

```
[47]:     User ID  Gender  Age  EstimatedSalary  Purchased
      0  15624510    Male   19            19000          0
      1  15810944    Male   35            20000          0
      2  15668575  Female   26            43000          0
      3  15603246  Female   27            57000          0
      4  15804002    Male   19            76000          0
```

```
[48]: data.Gender.value_counts()
```

```
[48]: Gender
      Female    204
      Male      196
      Name: count, dtype: int64
```

```
[68]: data=pd.get_dummies(data,columns=["Gender"],dtype=int,drop_first=True)
```

```
[69]: data
```

```
[69]:       User ID  Age  EstimatedSalary  Purchased  Gender_Male
      0    15624510   19            19000          0            1
      1    15810944   35            20000          0            1
      2    15668575   26            43000          0            0
      3    15603246   27            57000          0            0
      4    15804002   19            76000          0            1
      ..        …   …              …          …          …
      395  15691863   46            41000          1            0
      396  15706071   51            23000          1            1
      397  15654296   50            20000          1            0
      398  15755018   36            33000          0            1
      399  15594041   49            36000          1            0

      [400 rows x 5 columns]
```

```
[50]: (data['Gender_Male']==1).sum()
      (data['Gender_Male']==0).sum()
```

```
[50]: 204
```

```
[51]: data
```

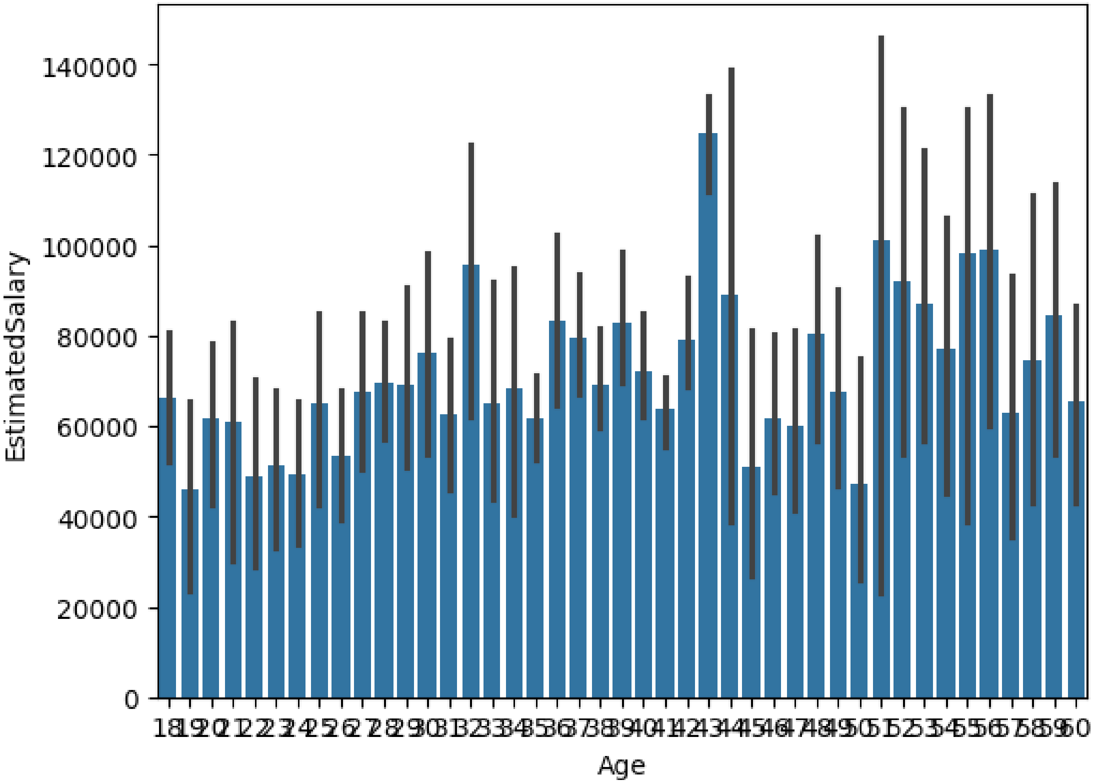
```
[51]:       User ID  Age  EstimatedSalary  Purchased  Gender_Male
      0    15624510   19            19000          0            1
      1    15810944   35            20000          0            1
      2    15668575   26            43000          0            0
      3    15603246   27            57000          0            0
      4    15804002   19            76000          0            1
      ..        …   …              …          …          …
```

```
395  15691863   46            41000          1             0
396  15706071   51            23000          1             1
397  15654296   50            20000          1             0
398  15755018   36            33000          0             1
399  15594041   49            36000          1             0

[400 rows x 5 columns]
```

[52]:
```
import seaborn as sns
sns.barplot(x="Age",y="EstimatedSalary",data=data)
```

[52]:
```
<Axes: xlabel='Age', ylabel='EstimatedSalary'>
```



[54]:
```
x=data.drop("Purchased",axis=1)
x
```

[54]:
```
      User ID  Age  EstimatedSalary  Gender_Male
0    15624510   19            19000            1
1    15810944   35            20000            1
2    15668575   26            43000            0
3    15603246   27            57000            0
4    15804002   19            76000            1
```

```
..         …   …              …          …
395  15691863   46          41000            0
396  15706071   51          23000            1
397  15654296   50          20000            0
398  15755018   36          33000            1
399  15594041   49          36000            0

[400 rows x 4 columns]
```

[55]:

```
x.shape
```

[55]:

```
(400, 4)
```

[56]:

```
y=data.drop("Gender_Male",axis=1)
y
```

[56]:

```
      User ID  Age  EstimatedSalary  Purchased
0    15624510   19            19000          0
1    15810944   35            20000          0
2    15668575   26            43000          0
3    15603246   27            57000          0
4    15804002   19            76000          0
..        …   …              …        …
395  15691863   46            41000          1
396  15706071   51            23000          1
397  15654296   50            20000          1
398  15755018   36            33000          0
399  15594041   49            36000          1

[400 rows x 4 columns]
```

[57]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
  ↪2,random_state=101)
```

[58]:

```
y.shape
```

[58]:

```
(400, 4)
```

[59]:

```
x.shape
```

[59]:

```
(400, 4)
```

[60]:

```
x_train.shape
```

[60]:

```
(320, 4)
```

[61]:

```
y_train.shape
```

[61]: (320, 4)

[62]:
```
from sklearn.linear_model import LogisticRegression
log_model=LogisticRegression()
log_model
```

[62]: LogisticRegression()

[71]:
```
data.head()
```

[71]:
```
     User ID  Age  EstimatedSalary  Purchased  Gender_Male
0   15624510   19            19000          0            1
1   15810944   35            20000          0            1
2   15668575   26            43000          0            0
3   15603246   27            57000          0            0
4   15804002   19            76000          0            1
```

[29]:
```
data.tail()
```

[29]:
```
       User ID  Age  EstimatedSalary  Purchased  Gender_Male
395   15691863   46            41000          1            0
396   15706071   51            23000          1            1
397   15654296   50            20000          1            0
398   15755018   36            33000          0            1
399   15594041   49            36000          1            0
```

[30]:
```
data.describe()
```

[30]:
```
            User ID         Age  EstimatedSalary   Purchased  Gender_Male
count  4.000000e+02  400.000000       400.000000  400.000000   400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500     0.490000
std    7.165832e+04   10.482877     34096.960282    0.479864     0.500526
min    1.556669e+07   18.000000     15000.000000    0.000000     0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000     0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000     0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000     1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000     1.000000
```

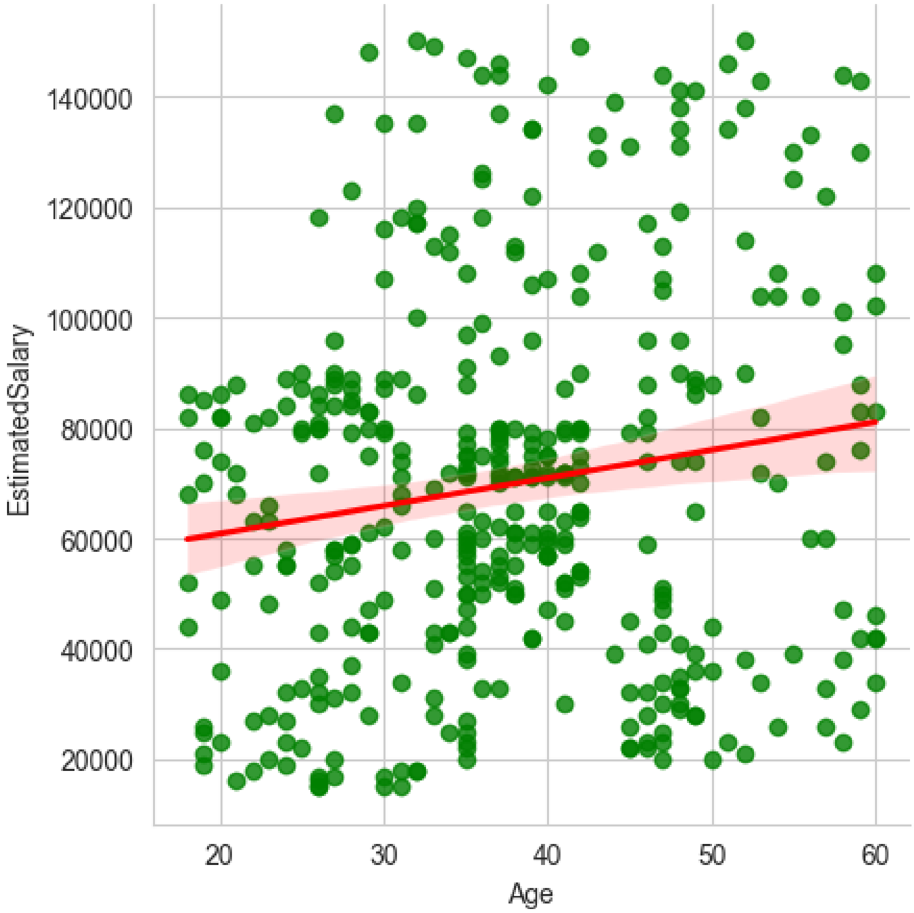
[31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪4,random_state=100)
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model
```

[31]: LogisticRegression()

[32]:
```
from sklearn.linear_model import Ridge
clf=Ridge()
clf.fit(x_train,y_train)
```

[32]:
```
Ridge()
```

[100]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lmplot(x="Age",y="EstimatedSalary",data=data,scatter_kws={"color":'green'␣
 ↪},line_kws={'color':"red"})
sns.set_style('whitegrid')
ax=plt.gca()
plt.gca()
plt.gca().set_facecolor('white')
```



[102]:
```
data=pd.get_dummies(data,columns=["Gender_Male"],dtype=int,drop_first=True)
data
```

```
[102]:       User ID  Age  EstimatedSalary  Purchased  Gender_Male_1
       0    15624510   19            19000          0              1
       1    15810944   35            20000          0              1
       2    15668575   26            43000          0              0
       3    15603246   27            57000          0              0
       4    15804002   19            76000          0              1
       ..        …   …              …        …            …
       395  15691863   46            41000          1              0
       396  15706071   51            23000          1              1
       397  15654296   50            20000          1              0
       398  15755018   36            33000          0              1
       399  15594041   49            36000          1              0

       [400 rows x 5 columns]
```

```
[104]: import numpy as np
```

```
[110]: #To check duplicate values
       duplicate_rows=data.duplicated()
       data[duplicate_rows].sum()
```

```
[110]: User ID            0
       Age                0
       EstimatedSalary    0
       Purchased          0
       Gender_Male_1      0
       dtype: int64
```

```
[111]: print("Before dropping duplicate:",data.shape)
       data.drop_duplicates()
       print("After dropping duplicate:",data.shape)
```

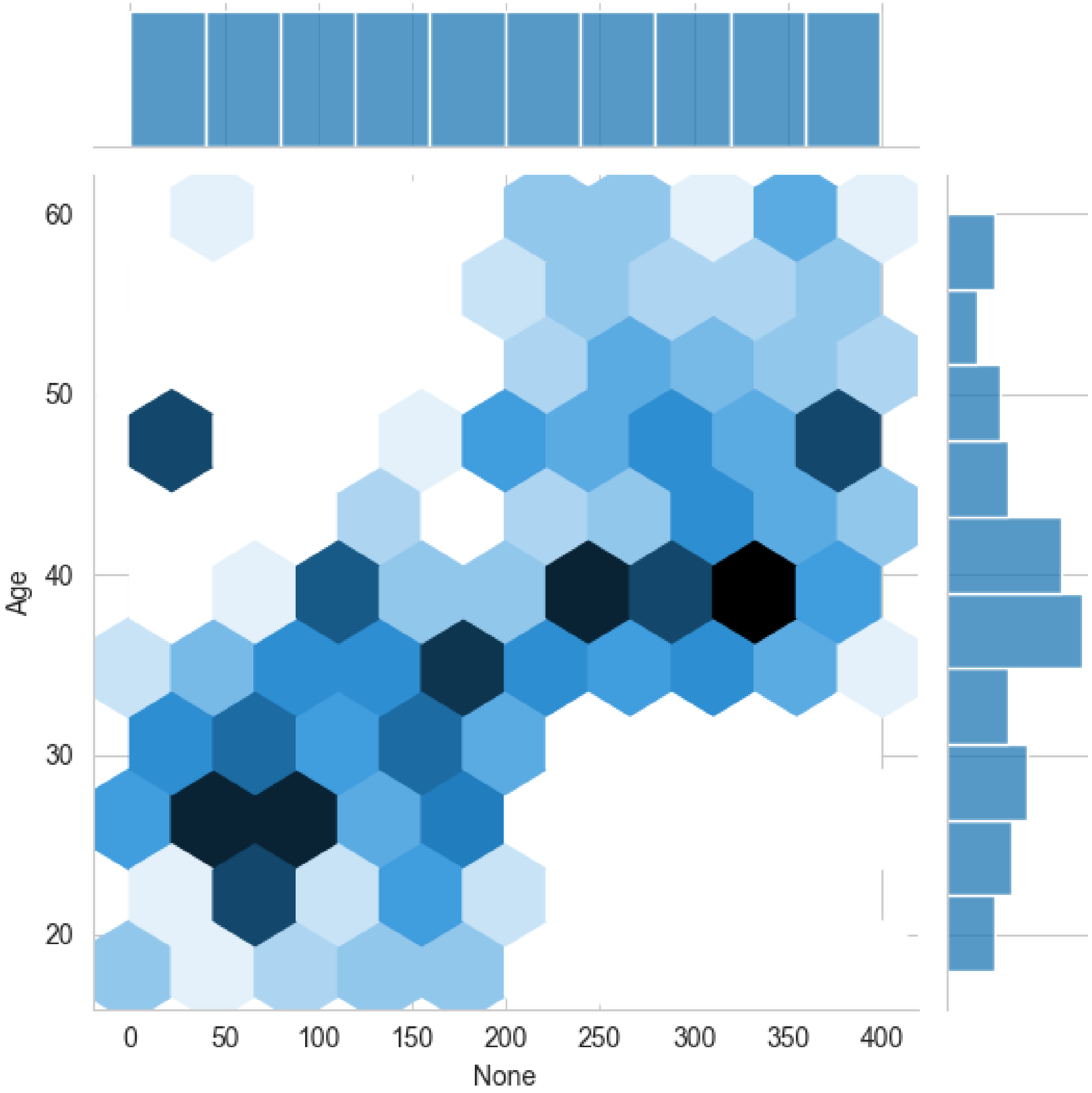
```
       Before dropping duplicate: (400, 5)
       After dropping duplicate: (400, 5)
```

```
[115]: # Based on index value try to check the performance
       response=data['Gender_Male_1']
       response.dtype
```

```
[115]: dtype('int32')
```

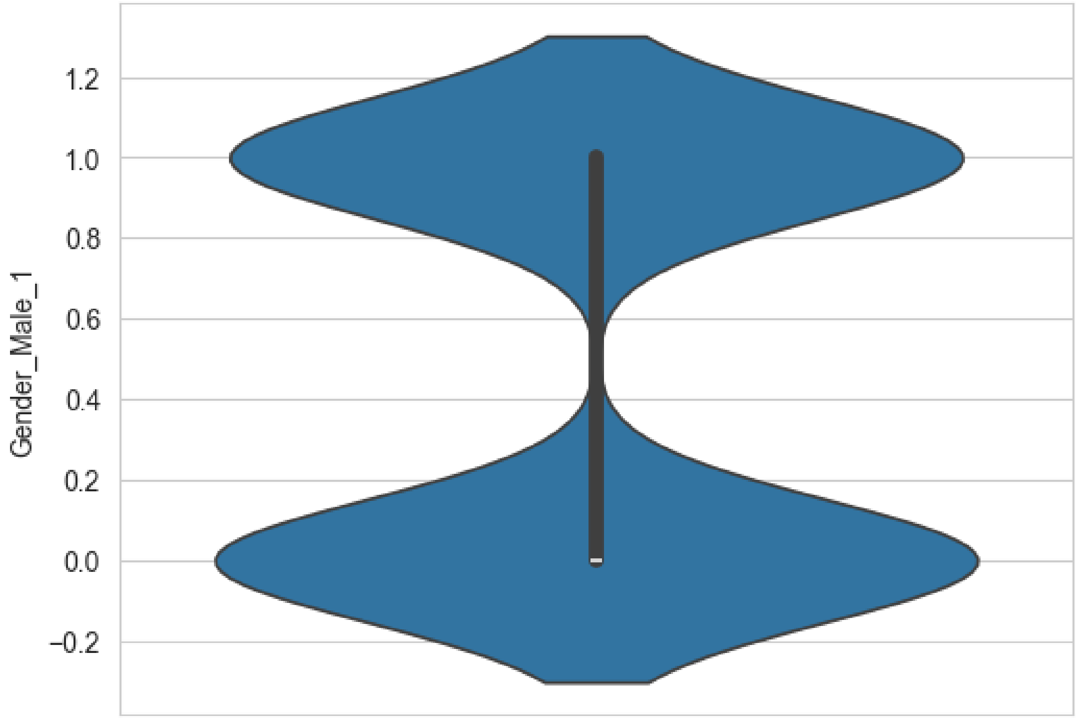
```
[125]: sns.jointplot(x=response.index,y="Age",data=data,kind='hex')
```

```
[125]: <seaborn.axisgrid.JointGrid at 0x2145ce6c3e0>
```

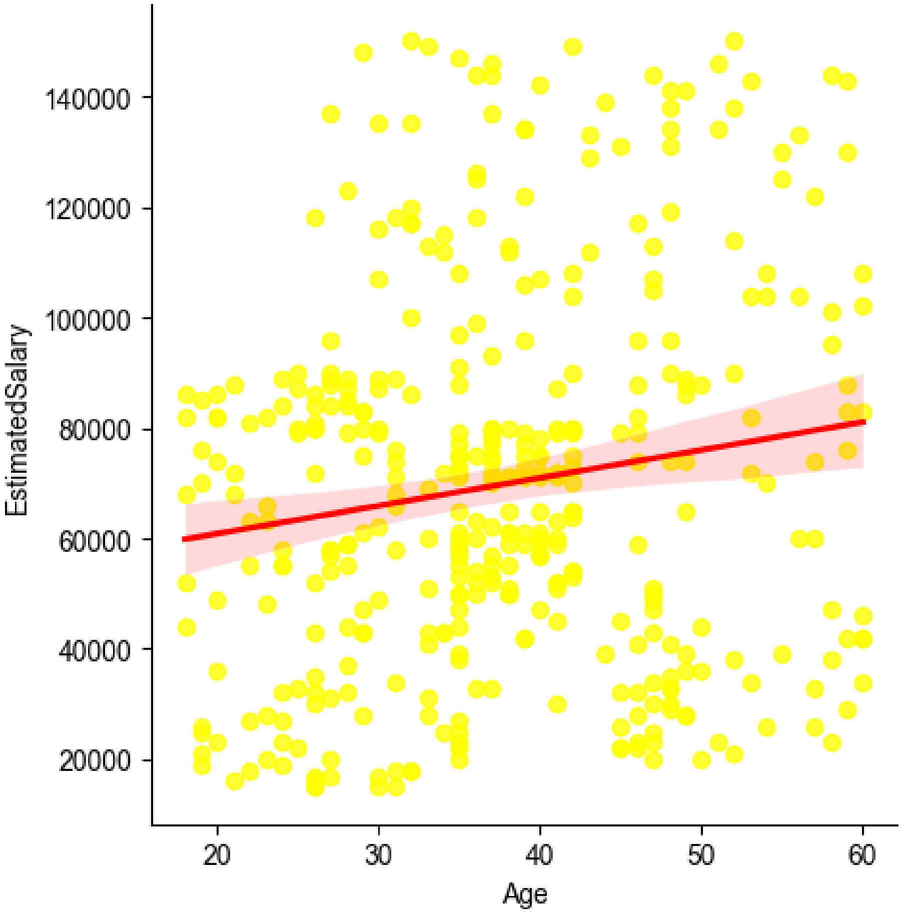


[126]:
```
sns.violinplot(response)
```

[126]:
```
<Axes: ylabel='Gender_Male_1'>
```



```
[65]: import seaborn as sns
      import matplotlib.pyplot as plt
      sns.lmplot(x="Age",y="EstimatedSalary",data=data,scatter_kws={"color":'yellow'␣
       ↪},line_kws={'color':"red"})
      sns.set_style('whitegrid')
      ax=plt.gca()
      plt.gca()
      plt.gca().set_facecolor('white')
```


140000
120000
100000
80000
60000
40000
20000
EstimatedSalary
20
30
40
50
60
Age

```
[ ]:
```