```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/TravelInsurancePrediction1.csv')
df
```

	Unnamed: 0	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelIn
0	0	31	Government Sector	Yes	400000	6	1	No	No	
1	1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	
2	2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	
3	3	28	Private Sector/Self Employed	Yes	700000	3	1	No	No	
4	4	28	Private Sector/Self Employed	Yes	700000	8	1	Yes	No	
...	...	...	...	...	...	...	...	...	...	...
1982	1982	33	Private Sector/Self Employed	Yes	1500000	4	0	Yes	Yes	
1983	1983	28	Private Sector/Self Employed	Yes	1750000	5	1	No	Yes	
1984	1984	28	Private Sector/Self Employed	Yes	1150000	6	1	No	No	
1985	1985	34	Private Sector/Self Employed	Yes	1000000	6	0	Yes	Yes	
1986	1986	34	Private Sector/Self Employed	Yes	500000	4	0	No	No	

1987 rows × 10 columns

```
df.isna().sum()
```

```
Unnamed: 0              0
Age                     0
Employment Type         0
GraduateOrNot           0
AnnualIncome            0
FamilyMembers           0
ChronicDiseases         0
FrequentFlyer          39
EverTravelledAbroad     0
TravelInsurance         0
dtype: int64
```

```
df.size
```

```
19870
```

```
df.shape
```

```
(1987, 10)
```

```
df.head()
```

	Unnamed: 0	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsur
0	0	31	Government Sector	Yes	400000	6	1	No	No	
1	1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	
2	2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	
3	3	28	Private	Yes	700000	3	1	No	No	

```
df.tail()
```

	Unnamed: 0	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelIn
1982	1982	33	Private Sector/Self Employed	Yes	1500000	4	0	Yes	Yes	
1983	1983	28	Private Sector/Self Employed	Yes	1750000	5	1	No	Yes	
1984	1984	28	Private Sector/Self Employed	Yes	1150000	6	1	No	No	
1985	1985	34	Private Sector/Self Employed	Yes	1000000	6	0	Yes	Yes	
1986	1986	34	Private Sector/Self Employed	Yes	500000	4	0	No	No	

```
df.dtypes
```

```
Unnamed: 0             int64
Age                    int64
Employment Type       object
GraduateOrNot         object
AnnualIncome           int64
FamilyMembers          int64
ChronicDiseases        int64
FrequentFlyer         object
EverTravelledAbroad   object
TravelInsurance        int64
dtype: object
```

```
df.columns
```

```
Index(['Unnamed: 0', 'Age', 'Employment Type', 'GraduateOrNot', 'AnnualIncome',
       'FamilyMembers', 'ChronicDiseases', 'FrequentFlyer',
       'EverTravelledAbroad', 'TravelInsurance'],
      dtype='object')
```

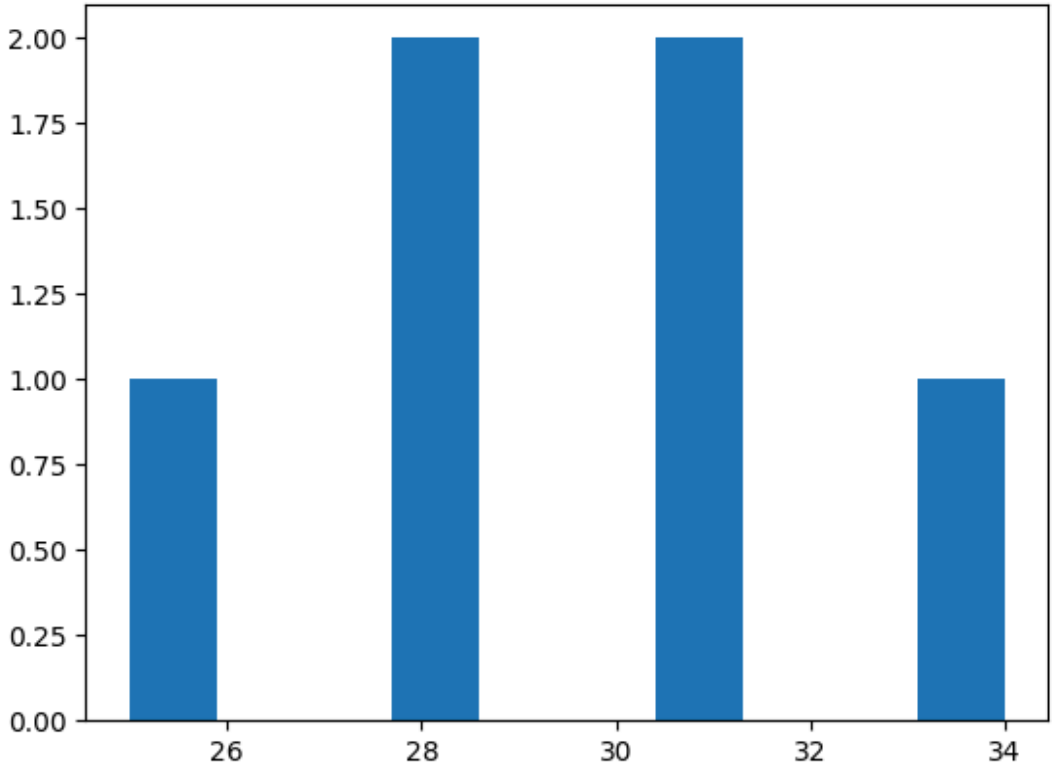
```
df["FrequentFlyer"]=df["FrequentFlyer"].fillna(df["FrequentFlyer"].mode().iloc[0])
df['FrequentFlyer'].fillna('0',inplace=True)
```

```
df.isna().sum()
```

```
Unnamed: 0            0
Age                   0
Employment Type       0
GraduateOrNot         0
AnnualIncome          0
FamilyMembers         0
ChronicDiseases       0
FrequentFlyer         0
EverTravelledAbroad   0
TravelInsurance       0
dtype: int64
```

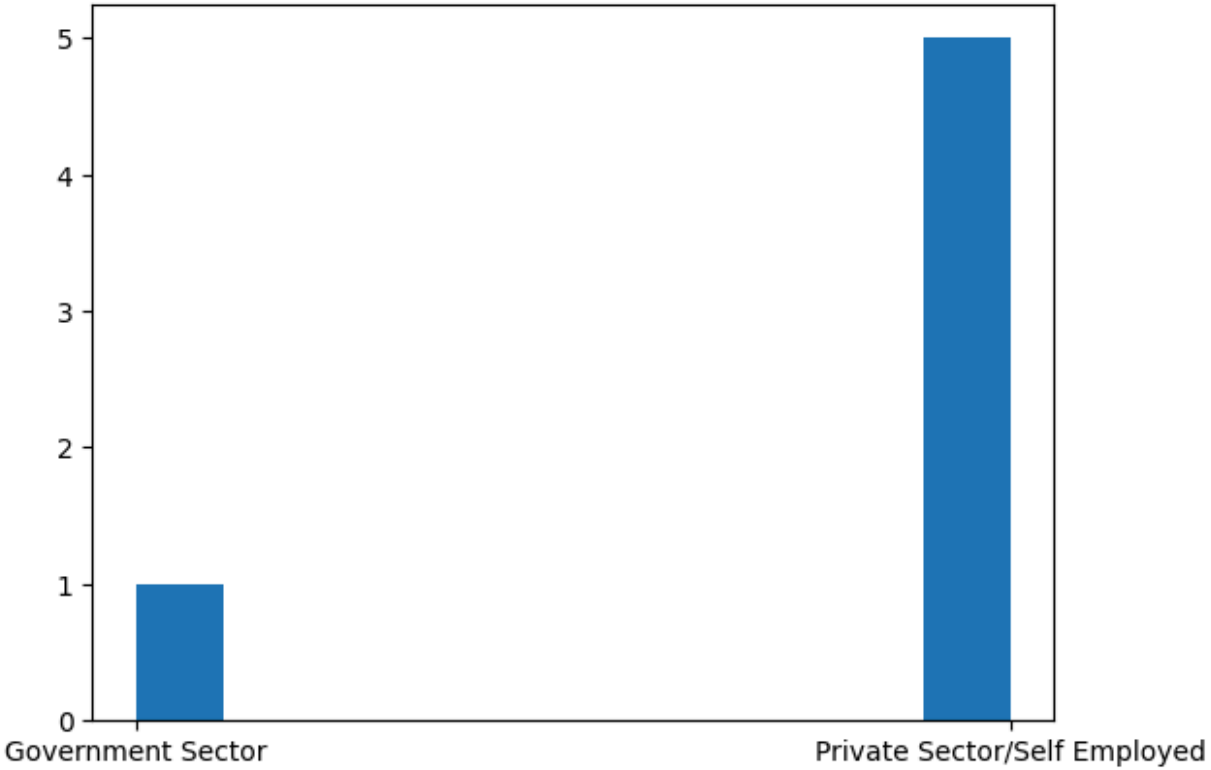
```
x=df['Age'].head(6)
plt.hist(x)
```

```
(array([1., 0., 0., 2., 0., 0., 2., 0., 0., 1.]),
 array([25. , 25.9, 26.8, 27.7, 28.6, 29.5, 30.4, 31.3, 32.2, 33.1, 34. ]),
 <BarContainer object of 10 artists>)
```




```
x=df['Employment Type'].head(6)
plt.hist(x)
```

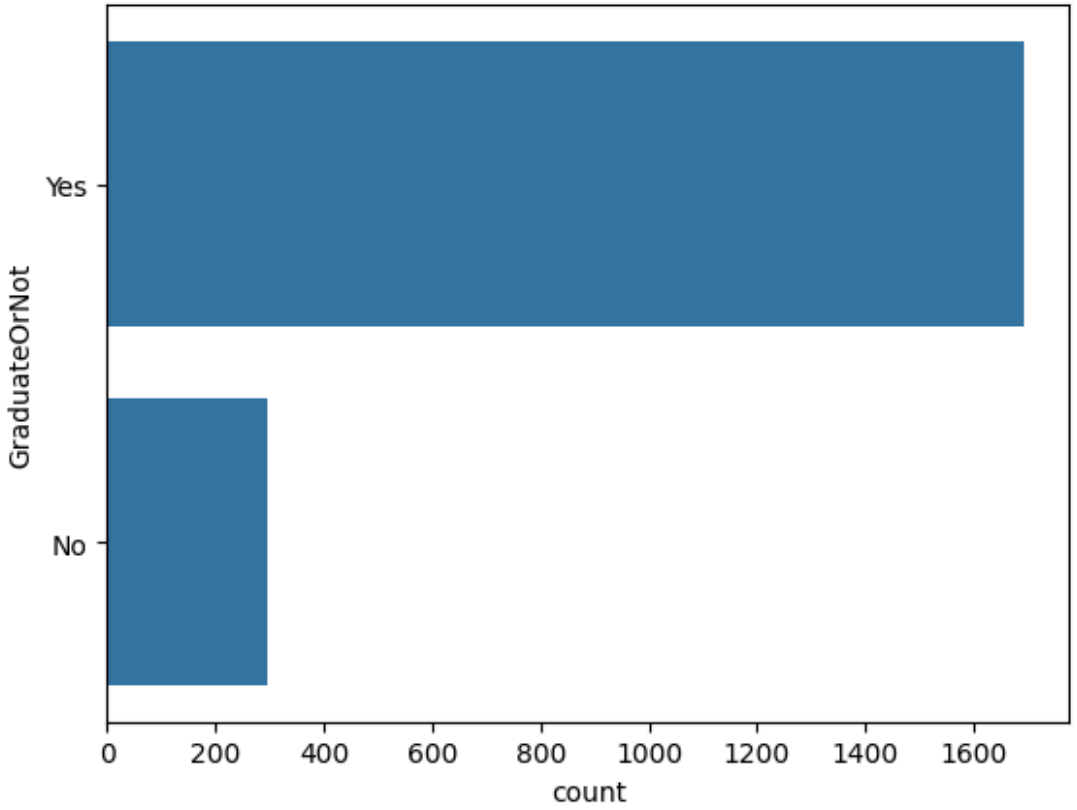
```
(array([1., 0., 0., 0., 0., 0., 0., 0., 0., 5.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



COUNT PLOT

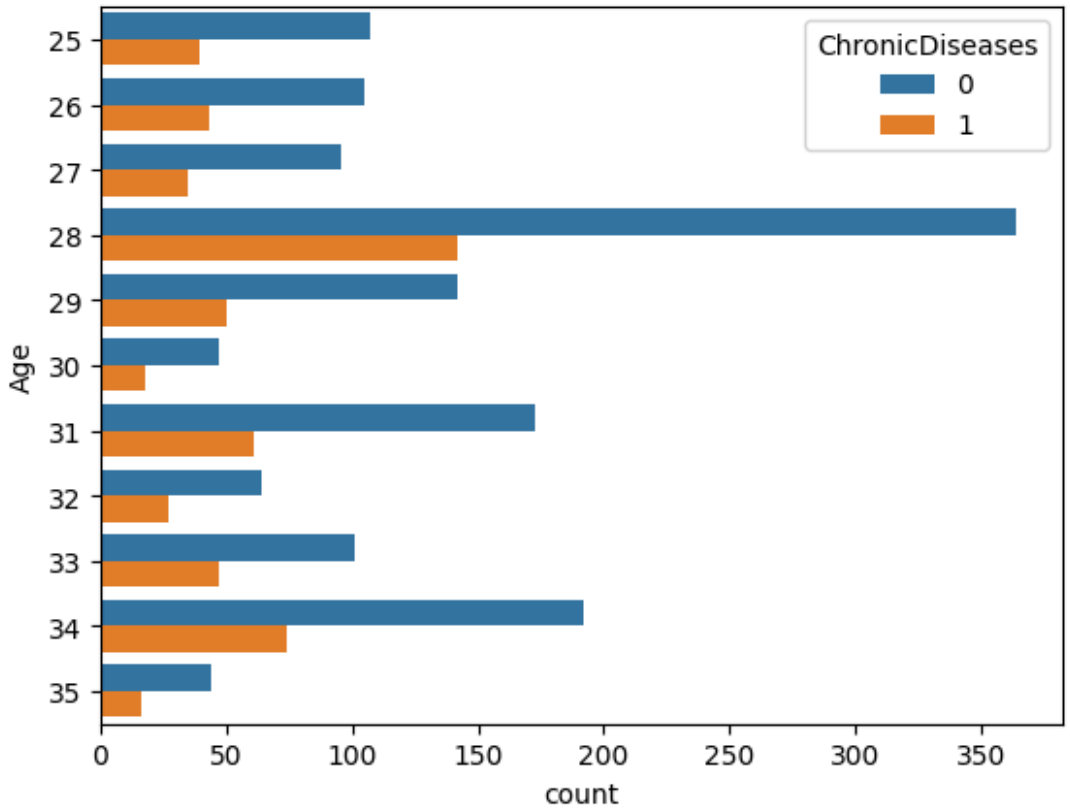
```
sns.countplot(y='GraduateOrNot',data=df)
```

```
<Axes: xlabel='count', ylabel='GraduateOrNot'>
```



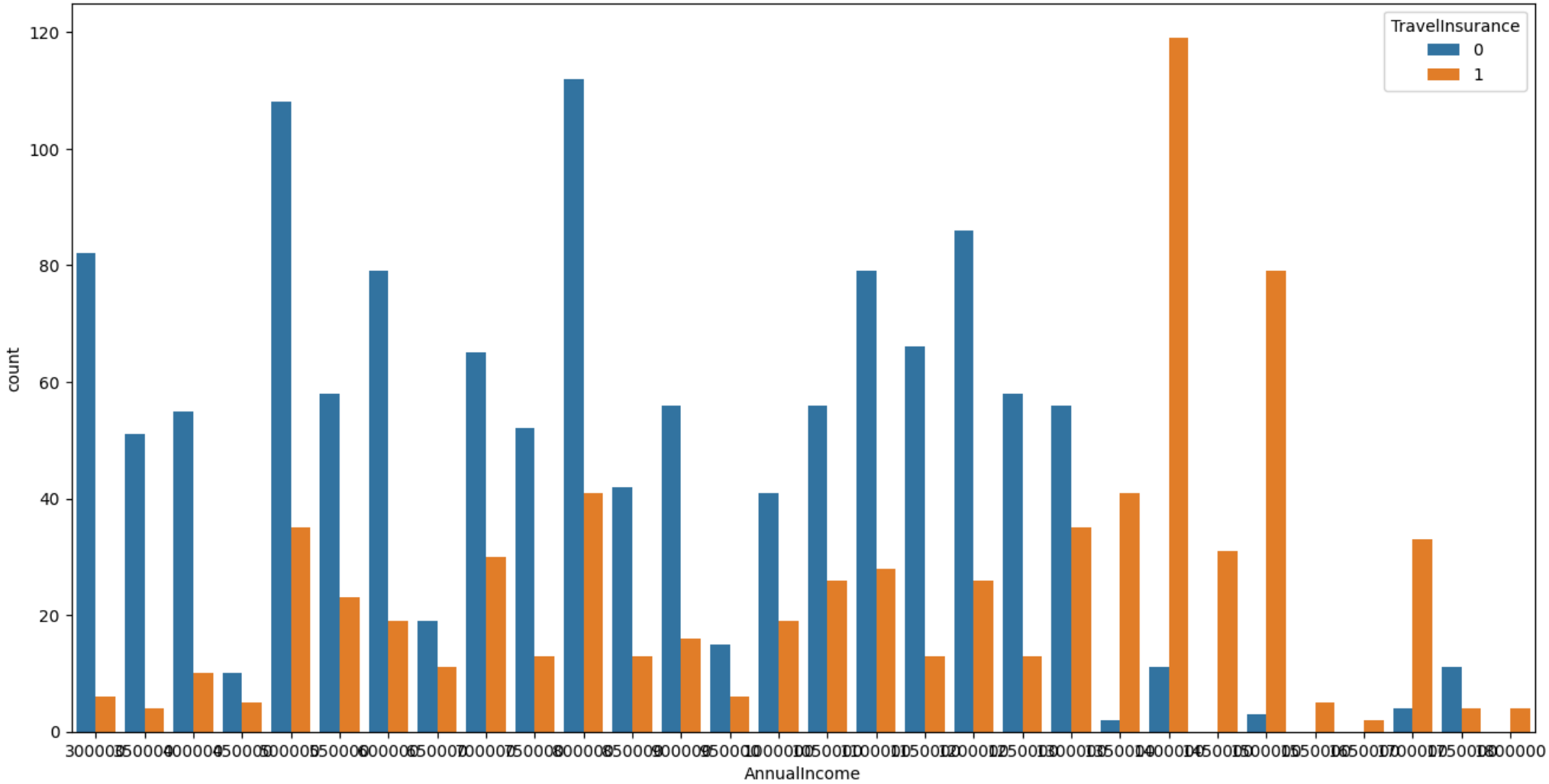

```
sns.countplot(y='Age',data=df,hue='ChronicDiseases')
```

```
<Axes: xlabel='count', ylabel='Age'>
```




```
plt.figure(figsize=(16,8))
sns.countplot(data=df, x=df['AnnualIncome'],hue=df['TravelInsurance'])
```

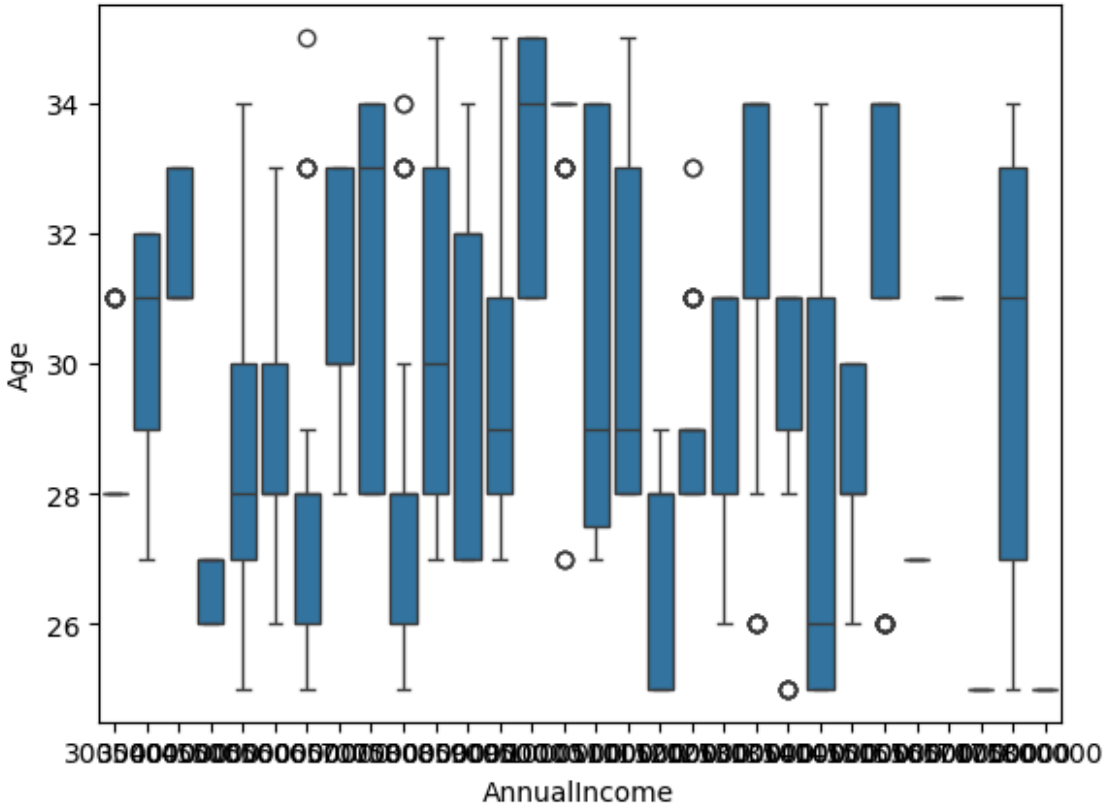
```
<Axes: xlabel='AnnualIncome', ylabel='count'>
```



BOX PLOT

```
sns.boxplot(y='Age',x='AnnualIncome',data=df)
```

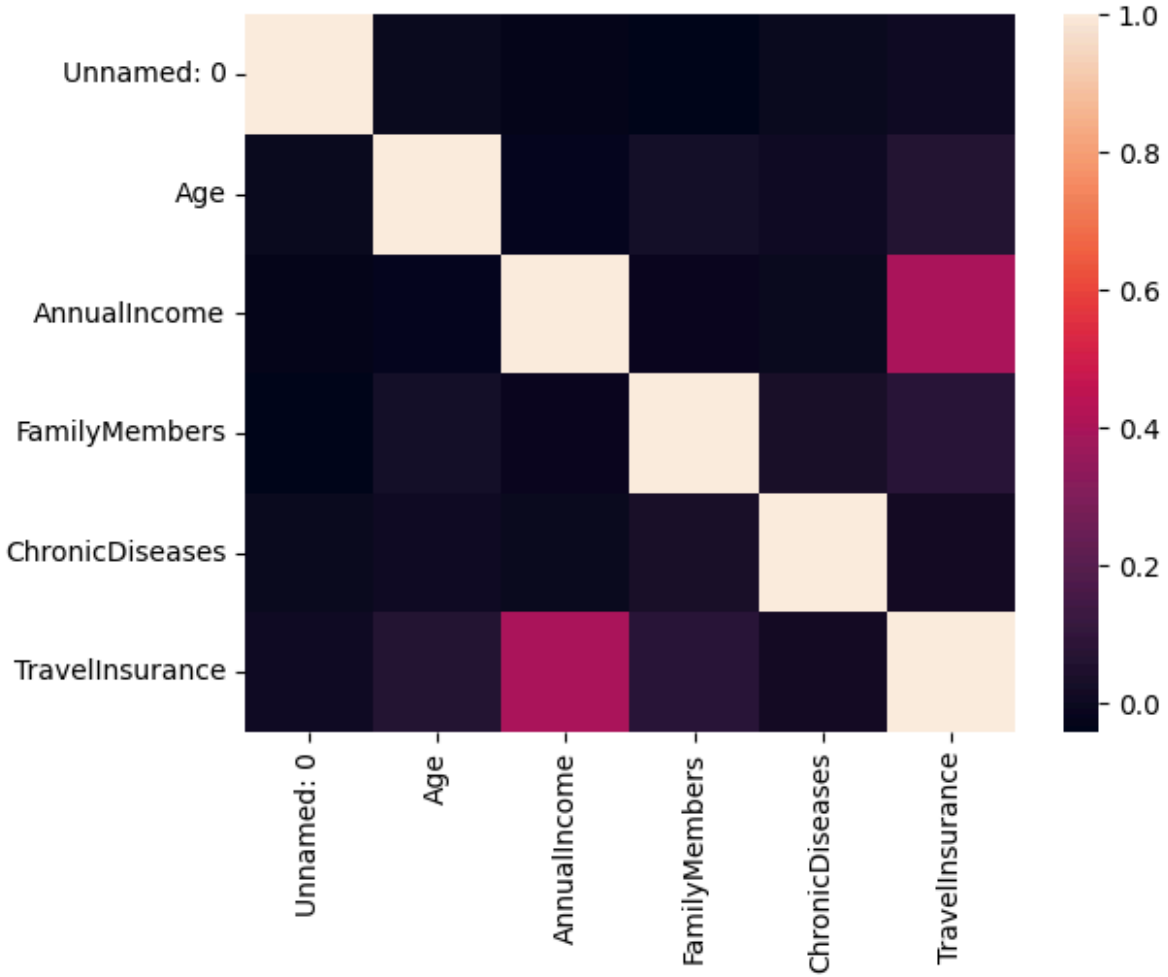
```
<Axes: xlabel='AnnualIncome', ylabel='Age'>
```



HEAT MAP

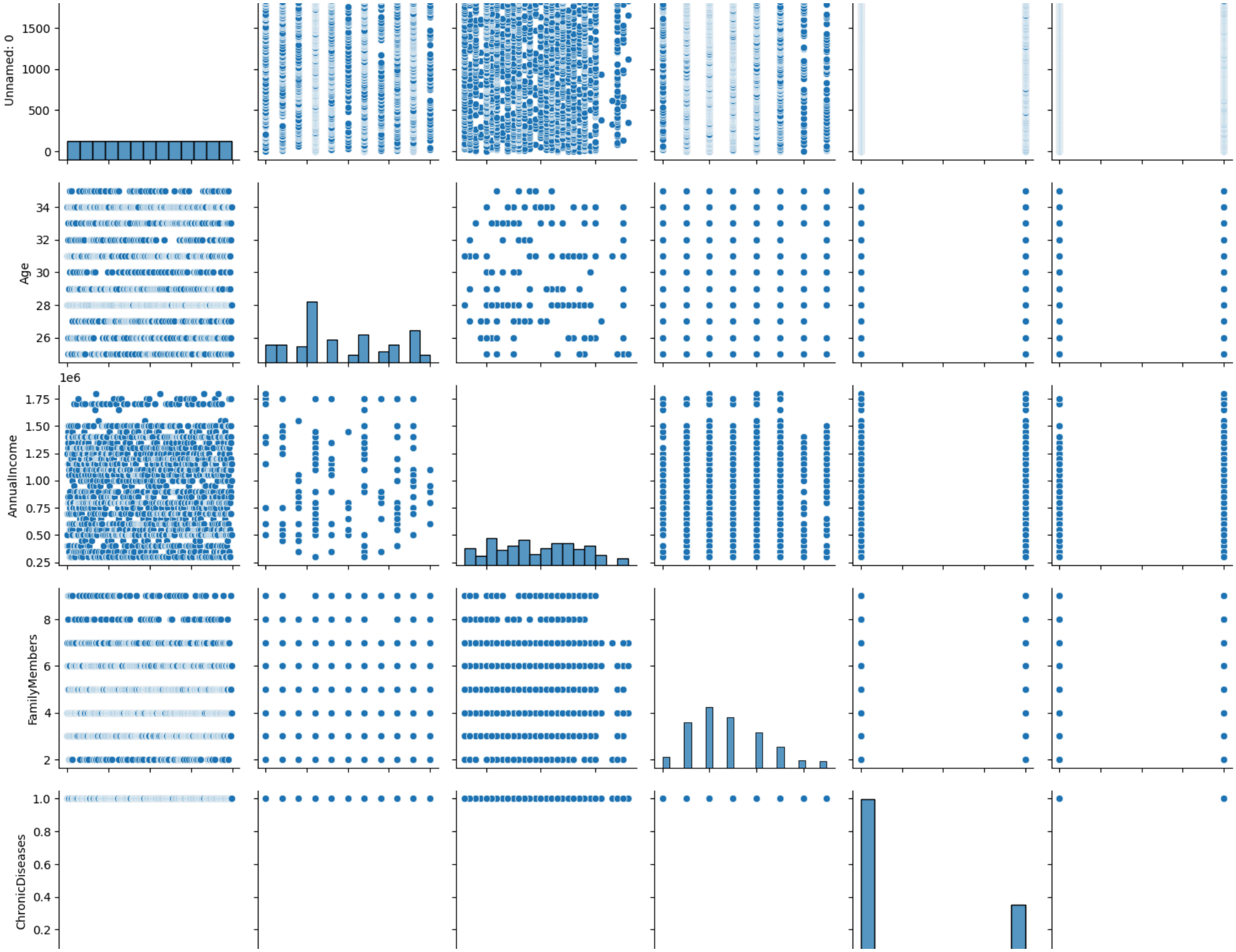
```
sns.heatmap(df.corr())
```

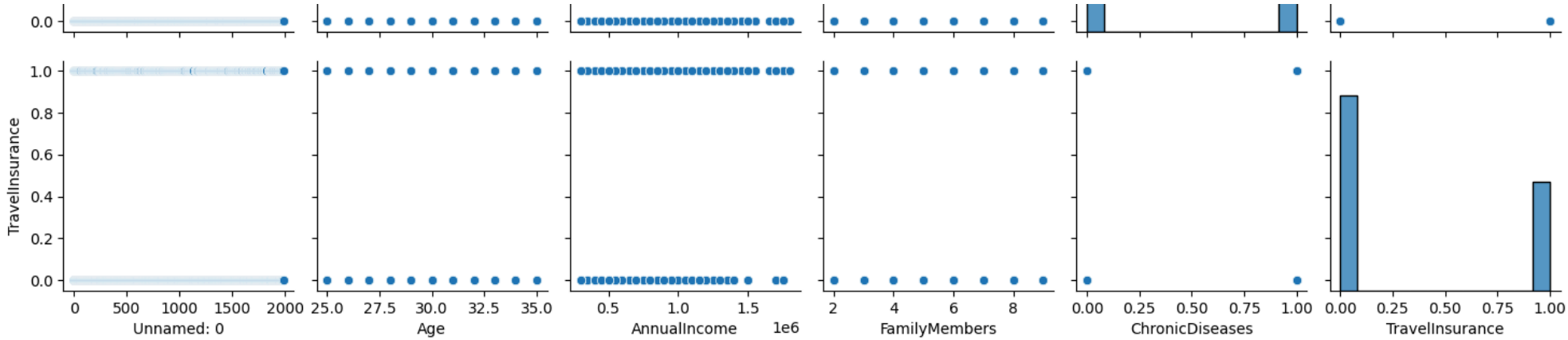
```
<ipython-input-84-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
  sns.heatmap(df.corr())
<Axes: >
```



PAIR PLOT

```
sns.pairplot(df)
```


Unnamed: 0
0
500
1000
1500
Age
26
28
30
32
34
1e6
AnnualIncome
0.25
0.50
0.75
1.00
1.25
1.50
1.75
FamilyMembers
2
4
6
8
ChronicDiseases
0.2
0.4
0.6
0.8
1.0




```
dummy=pd.get_dummies(df[['Employment Type', 'GraduateOrNot','FrequentFlyer',
       'EverTravelledAbroad']],drop_first=True)
dummy
```

	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelledAbroad_Yes
0	0	1	0	0
1	1	1	0	0
2	1	1	0	0
3	1	1	0	0
4	1	1	1	0
...	...	...	...	...
1982	1	1	1	1
1983	1	1	0	1
1984	1	1	0	0
1985	1	1	1	1
1986	1	1	0	0

1987 rows × 4 columns

```
df2=pd.concat([df,dummy],axis=1)
df2
```

	Unnamed: 0	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelIn
0	0	31	Government Sector	Yes	400000	6	1	No	No	
1	1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	
2	2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	
3	3	28	Private Sector/Self Employed	Yes	700000	3	1	No	No	
4	4	28	Private Sector/Self Employed	Yes	700000	8	1	Yes	No	
...	...	...	...	...	...	...	...	...	...	
1982	1982	33	Private Sector/Self Employed	Yes	1500000	4	0	Yes	Yes	
1983	1983	28	Private Sector/Self Employed	Yes	1750000	5	1	No	Yes	
1984	1984	28	Private Sector/Self Employed	Yes	1150000	6	1	No	No	
1985	1985	34	Private Sector/Self Employed	Yes	1000000	6	0	Yes	Yes	
1986	1986	34	Private Sector/Self Employed	Yes	500000	4	0	No	No	

1987 rows × 14 columns

1987 rows × 14 columns

```
df2=df2.drop(['Unnamed: 0','Employment Type', 'GraduateOrNot','FrequentFlyer','EverTravelledAbroad'],axis=1)
df2
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	TravelInsurance	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelle
0	31	400000	6	1	0	0	1	0	
1	31	1250000	7	0	0	1	1	0	
2	34	500000	4	1	1	1	1	0	
3	28	700000	3	1	0	1	1	0	
4	28	700000	8	1	0	1	1	1	
...	...	...	...	...	...	...	...	...	
1982	33	1500000	4	0	1	1	1	1	
1983	28	1750000	5	1	0	1	1	0	
1984	28	1150000	6	1	0	1	1	0	
1985	34	1000000	6	0	1	1	1	1	
1986	34	500000	4	0	0	1	1	0	

1987 rows × 9 columns

```
df2
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	TravelInsurance	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelle
0	31	400000	6	1	0	0	1	0	
1	31	1250000	7	0	0	1	1	0	
2	34	500000	4	1	1	1	1	0	
3	28	700000	3	1	0	1	1	0	
4	28	700000	8	1	0	1	1	1	
...	...	...	...	...	...	...	...	...	
1982	33	1500000	4	0	1	1	1	1	
1983	28	1750000	5	1	0	1	1	0	
1984	28	1150000	6	1	0	1	1	0	
1985	34	1000000	6	0	1	1	1	1	
1986	34	500000	4	0	0	1	1	0	

1987 rows × 9 columns

```
df2.dtypes
```

```
Age                                          int64
AnnualIncome                                 int64
FamilyMembers                                int64
ChronicDiseases                              int64
TravelInsurance                              int64
Employment Type_Private Sector/Self Employed    uint8
GraduateOrNot_Yes                            uint8
FrequentFlyer_Yes                            uint8
EverTravelledAbroad_Yes                      uint8
dtype: object
```

```
x=df2.drop(['TravelInsurance'],axis=1)
x
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelledAbroad_Yes
0	31	400000	6	1	0	1	0	0
1	31	1250000	7	0	1	1	0	0
2	34	500000	4	1	1	1	0	0
3	28	700000	3	1	1	1	0	0
4	28	700000	8	1	1	1	1	0
...	...	...	...	...	...	...	...	...
1982	33	1500000	4	0	1	1	1	1
1983	28	1750000	5	1	1	1	0	1
1984	28	1150000	6	1	1	1	0	0
1985	34	1000000	6	0	1	1	1	1
1986	34	500000	4	0	1	1	0	0

```
y=df2['TravelInsurance']
y
```

```
0       0
1       0
2       1
3       0
4       0
       ..
1982    1
1983    0
1984    0
1985    1
1986    0
Name: TravelInsurance, Length: 1987, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42)
x_train
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelledAbroad_Yes
710	26	1450000	3	1	0	1	1	1
170	31	1350000	6	1	1	1	0	1
292	27	850000	4	0	0	1	0	0
1675	33	750000	7	1	1	1	0	0
579	33	400000	3	0	0	1	0	0
...	...	...	...	...	...	...	...	...
1130	28	550000	5	0	1	1	1	0
1294	28	600000	4	0	1	1	0	0
860	34	1000000	4	1	1	1	0	0
1459	34	1500000	5	1	1	1	1	1
1126	29	1200000	4	0	1	1	0	0

```
x_test
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	Employment Type_Private Sector/Self Employed	GraduateOrNot_Yes	FrequentFlyer_Yes	EverTravelledAbroad_Yes
212	28	750000	5	1	1	1	0	0
1517	26	1400000	4	1	1	1	0	1
785	29	1200000	7	0	1	1	0	1
1175	33	400000	4	1	0	1	0	0
1760	34	1500000	3	0	1	1	1	1
...	...	...	...	...	...	...	...	...
554	32	800000	6	0	0	1	0	0
1198	33	400000	4	0	0	1	0	0
1878	28	700000	4	0	1	1	0	0
182	35	800000	3	0	1	0	1	0
1320	31	1350000	3	0	1	1	1	1

```
y_train
```

```
710     1
170     1
292     0
1675    1
579     0
       ..
1130    0
1294    0
860     0
1459    1
1126    0
Name: TravelInsurance, Length: 1490, dtype: int64
```

```
y_test
```

```
212     0
1517    1
785     0
1175    0
1760    1
       ..
554     0
1198    0
1878    1
182     1
1320    1
Name: TravelInsurance, Length: 497, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
```

StandardScaler

StandardScaler()

```
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

SVC MODEL CREATION

```
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
```

SVC

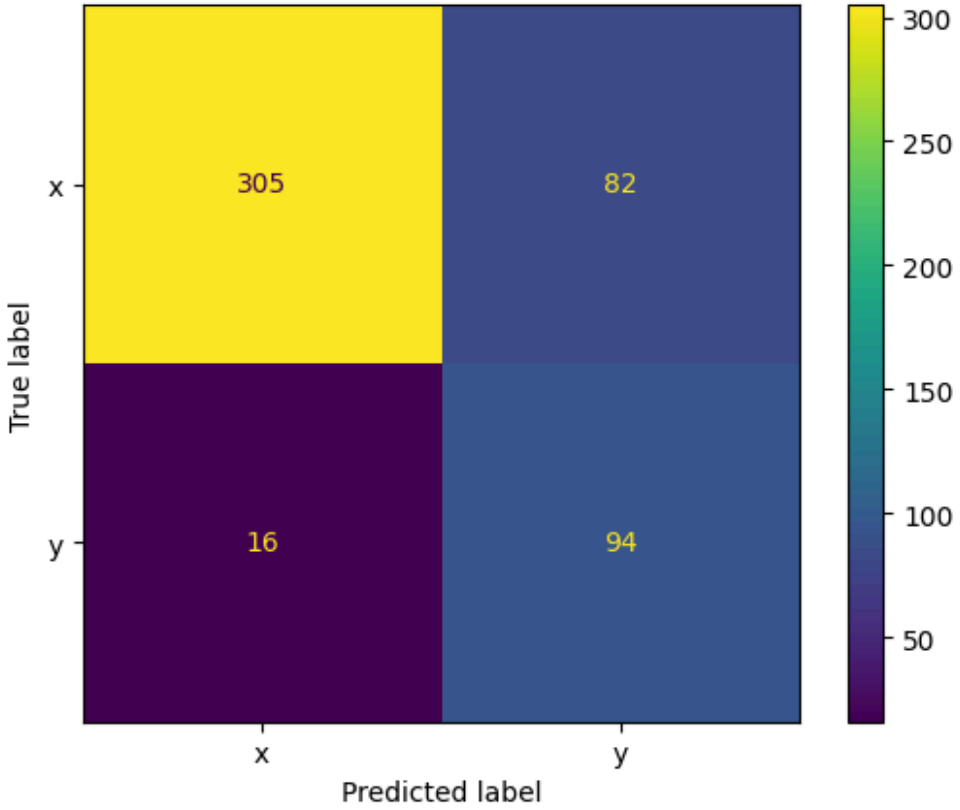
SVC()

```
y_pred=model.predict(x_test)
y_pred
```

```
array([0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 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, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 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, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       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, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,ConfusionMatrixDisplay
mat=confusion_matrix(y_pred,y_test)
label=['x','y']
cmd=ConfusionMatrixDisplay(mat,display_labels=label)
cmd.plot()
print(mat)
```

```
[[305  82]
 [ 16  94]]
```




```
score=accuracy_score(y_pred,y_test)
score
```

```
0.8028169014084507
```

```
report=classification_report(y_pred,y_test)
report
```

```
'              precision    recall  f1-score   support\n\n           0       0.95      0.79      0.86       387\n           1       0.53      0.85      0.66       110\n\n    accuracy                           0.80       497\n   macro avg       0.74      0.82      0.76       497\nweighted avg       0.86      0.80      0.82       497\n'
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
km=KNeighborsClassifier(n_neighbors=7)
```

```
km.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

```
y_pred=km.predict(x_test)
y_pred
```

```
array([0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 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, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

```
mat=confusion_matrix(y_pred,y_test)
mat
```

```
array([[291,  89],
       [ 30,  87]])
```

```
score=accuracy_score(y_pred,y_test)
score
```

```
0.7605633802816901
```

NAIVE BAYES

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
```

```
GaussianNB()
```