**import** pandas **as** pd

2

**import** seaborn **as** sns

3

**%**matplotlib inline

4

**import** matplotlib.pyplot **as** plt

In [2]:

1

data **=** pd.read\_csv("D:/Top\_mentor/Nov/5th\_nov/KNN\_Project\_5&6/KNN Assignment/iphone\_purchase\_records.csv")

In [3]:

1

df1**=**data.copy()

In [4]:

1

df1.head()

. . .

In [5]:

1

print('No. of rows', df1.shape[0])

No. of rows 400

In [6]:

1

print('No. of columns', df1.shape[1])

No. of columns 4

In [7]:

1

df1.columns

Out[7]:

Index(['Gender', 'Age', 'Salary', 'Purchase Iphone'], dtype='object')

**Total number of Male and Female**

In [8]:

1

gen**=**df1['Gender'].value\_counts()

2

print('Gender distribution',gen)

Gender distribution Female 204

Male 196

Name: Gender, dtype: int64

In [80]:

1

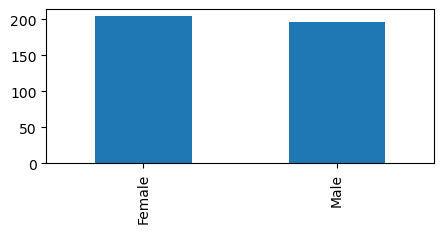
plt.figure(figsize**=**(5,2))

2

df1['Gender'].value\_counts().plot(kind**=**'bar')

3

plt.show()



**Total number of male have iphone**

In [10]:

1

male\_i **=** len(df1[(df1['Gender']**==**'Male')**&**(df1['Purchase Iphone']**==**1)])

2

print('Total number of male have iphone', male\_i)

Total number of male have iphone 66

**Total number of female have iphone**

In [11]:

1

female\_i **=** len(df1[(df1['Gender']**==**'Female')**&**(df1['Purchase Iphone']**==**1)])

2

print('Total number of male have iphone', female\_i)

Total number of male have iphone 77

**Total number of male belongs age (15 - 40) have iphone**

In [12]:

1

len(df1[(df1['Age'].between(15,40)) **&** (df1['Gender']**==**'Male')**&**(df1['Purchase Iphone']**==**1)])

Out[12]:

21

**Total number of female belongs age (15 - 40) have iphone**

In [13]:

1

len(df1[(df1['Age'].between(15,40)) **&** (df1['Gender']**==**'Female')**&**(df1['Purchase Iphone']**==**1)])

Out[13]:

18

**Maimum salary range by gender, who have iphone**

In [14]:

1

df1.groupby(['Salary'])[['Gender','Purchase Iphone']].max().sort\_values(by**=**'Salary',ascending**=False**).head()

Out[14]:

|  | **Gender** | **Purchase Iphone** |
| --- | --- | --- |
| **Salary** |  |  |
| **150000** | Male | 1 |
| **149000** | Male | 1 |
| **148000** | Male | 1 |
| **147000** | Female | 1 |
| **146000** | Female | 1 |

**Total no male have iphone salary range (20000-50000)**

In [31]:

1

m **=** sum((df1['Gender']**==** 'Male') **&** (df1['Salary'].between(20000,50000)))

2

print('Total no male have iphone salary range (20000-50000)',m)

Total no male have iphone salary range (20000-50000) 47

**Total no female have iphone salary range (20000-50000)**

In [33]:

1

f **=** sum((df1['Gender']**==** 'Female') **&** (df1['Salary'].between(20000,50000)))

2

print('Total no male have iphone salary range (20000-50000)',f)

Total no male have iphone salary range (20000-50000) 62

**Mimmum salary by gender**

In [35]:

1

df1.groupby('Gender')['Salary'].min()

Out[35]:

Gender

Female 15000

Male 15000

Name: Salary, dtype: int64

**Age and salary distribution by gender**

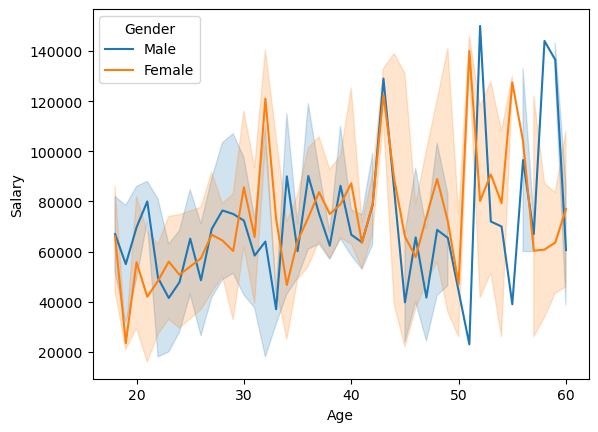
In [84]:

1

sns.lineplot(x**=**'Age', y**=**'Salary', hue**=**'Gender', data**=**df1)

2

plt.show()



**iphone holder analysis by age and gender**

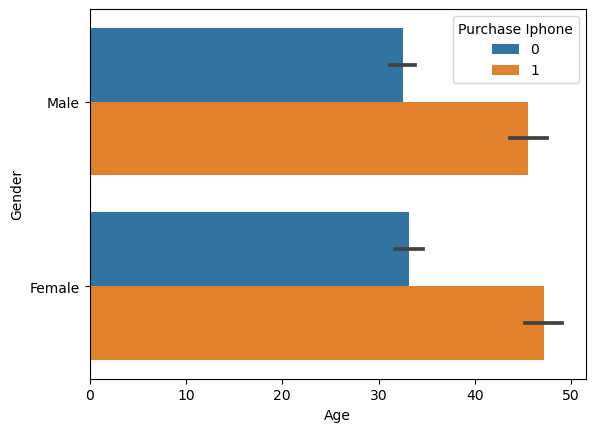
In [77]:

1

sns.barplot(x**=**'Age', y**=**'Gender', hue**=**'Purchase Iphone',data**=**df1)

2

plt.show()



**Total number of iphone holders**

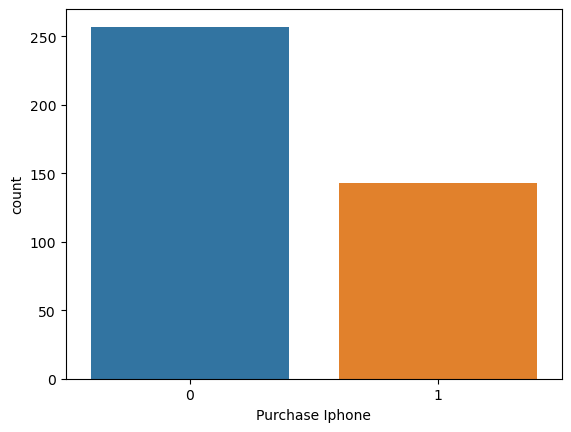
In [76]:

1

sns.countplot(x**=**'Purchase Iphone',data**=**df1)

2

plt.show()



**Average salary by gender**

In [203]:

1

round(df1.groupby('Gender')['Salary'].mean())

Out[203]:

Gender

Female 71760.0

Male 67643.0

Name: Salary, dtype: float64

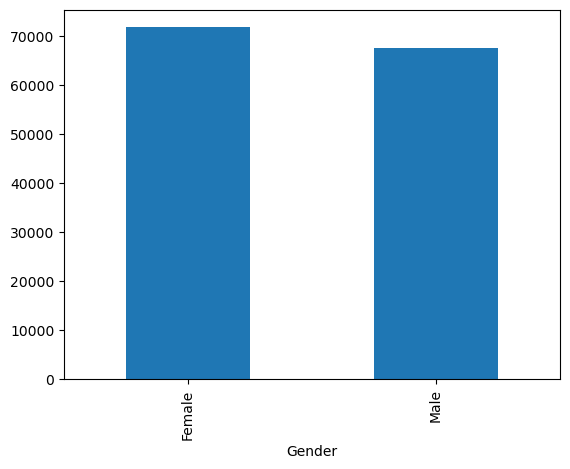
In [75]:

1

df1.groupby('Gender')['Salary'].mean().plot(kind**=**'bar')

2

plt.show()



**Don't have iphone oriented with age group 25 to 40**

In [100]:

1

null\_i**=**len(df1[(df1['Age'].between(25,40))**&**(df1['Purchase Iphone']**==**0)])

2

print("Don't have iphone oriented with age group 25 to 40", null\_i)

Don't have iphone oriented with age group 25 to 40 171

**Correlation**

In [105]:

1

df1.corr()

C:\Users\Acer\AppData\Local\Temp\ipykernel\_13048\473017434.py:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

df1.corr()

Out[105]:

|  | **Age** | **Salary** | **Purchase Iphone** |
| --- | --- | --- | --- |
| **Age** | 1.000000 | 0.155238 | 0.622454 |
| **Salary** | 0.155238 | 1.000000 | 0.362083 |
| **Purchase Iphone** | 0.622454 | 0.362083 | 1.000000 |

In [114]:

1

df1.isnull().sum()

Out[114]:

Gender 0

Age 0

Salary 0

Purchase Iphone 0

dtype: int64

**KNN Model**

In [115]:

1

dummy **=** pd.get\_dummies(df1['Gender']).astype('int')

2

dummy

. . .

In [116]:

1

df2 **=** pd.concat([df1,dummy], axis**=**1)

In [117]:

1

df2.head()

. . .

In [143]:

1

X**=** df2[['Female','Male','Age','Salary']]

2

​

In [145]:

1

y**=**df2['Purchase Iphone']

2

​

In [146]:

1

**from** sklearn.model\_selection **import** train\_test\_split

In [147]:

1

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,train\_size **=** 0.8)

In [148]:

1

**from** sklearn.neighbors **import** KNeighborsClassifier

**Identifiaction of K**

In [152]:

1

**from** math **import** sqrt

2

sqrt(df2.shape[0])

Out[152]:

20.0

In [190]:

1

clf**=** KNeighborsClassifier(n\_neighbors**=**1)

2

clf.fit(X\_train, y\_train)

Out[190]:

KNeighborsClassifier

KNeighborsClassifier(n\_neighbors=1)

In [191]:

1

pred **=** clf.predict(X\_test)

2

pred

Out[191]:

array([0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,

0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,

0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,

0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0], dtype=int64)

**Check the accuracy of the model**

In [192]:

1

**from** sklearn.metrics **import** accuracy\_score, confusion\_matrix

In [193]:

1

print('Accuracy level = ',accuracy\_score(y\_test,pred)**\***100)

Accuracy level = 82.5

In [194]:

1

confusion\_matrix(y\_test, pred)

Out[194]:

array([[48, 11],

[ 3, 18]], dtype=int64)

In [195]:

1

sns.heatmap(confusion\_matrix(y\_test, pred), annot **=True**)

. . .

In [196]:

1

error **=** []

In [197]:

1

**for** i **in** range(1,20):

2

clf**=**KNeighborsClassifier(n\_neighbors**=**i)

3

clf.fit(X\_train,y\_train)

4

predict\_i **=** clf.predict(X\_test)

5

error.append (1**-**accuracy\_score(y\_test,predict\_i))

In [185]:

1

error

. . .

In [ ]: