assignment-5

February 19, 2024

[21]:
```
import pandas as pd
import numpy as np
```

[22]:
```
df = pd.read_csv("/content/Social_Network_Ads.csv")
```

[23]:
```
df.head()
```

[23]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19.0	19000.0	0
1	15810944	Male	35.0	20000.0	0
2	15668575	Female	26.0	43000.0	0
3	15603246	Female	27.0	57000.0	0
4	15804002	Male	19.0	76000.0	0

[24]:
```
df.info()
```

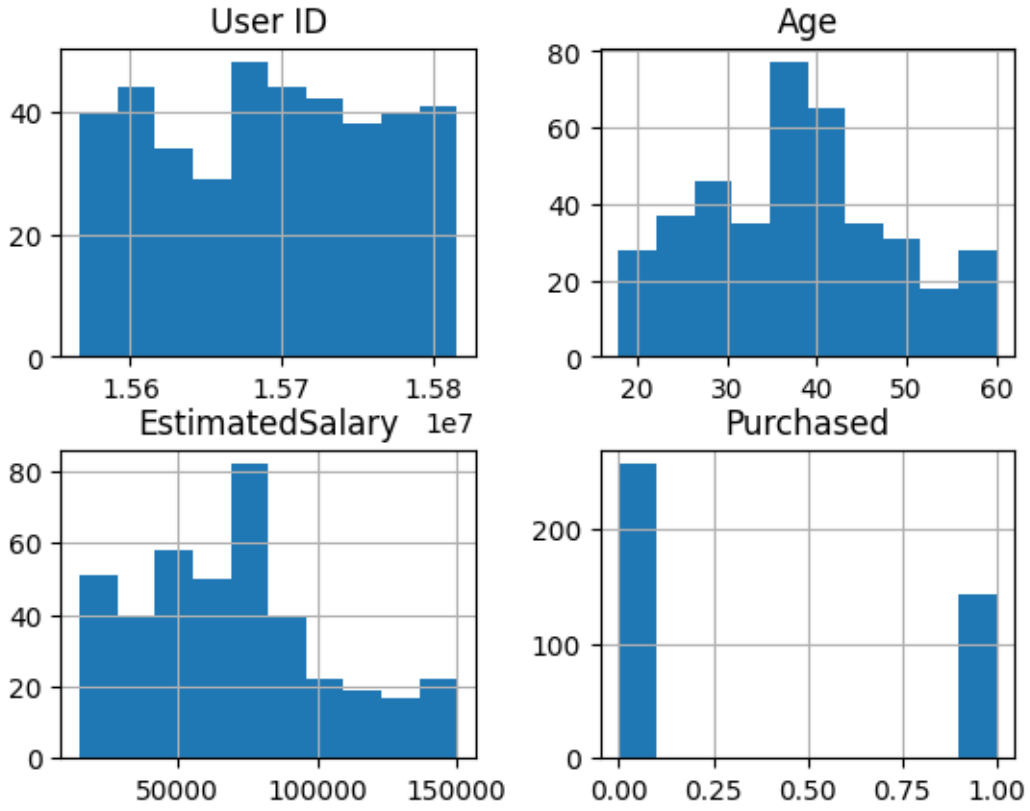
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    float64
 3   EstimatedSalary  400 non-null    float64
 4   Purchased        400 non-null    int64
dtypes: float64(2), int64(2), object(1)
memory usage: 15.8+ KB
```

[25]:
```
df.isnull().sum()
```

[25]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

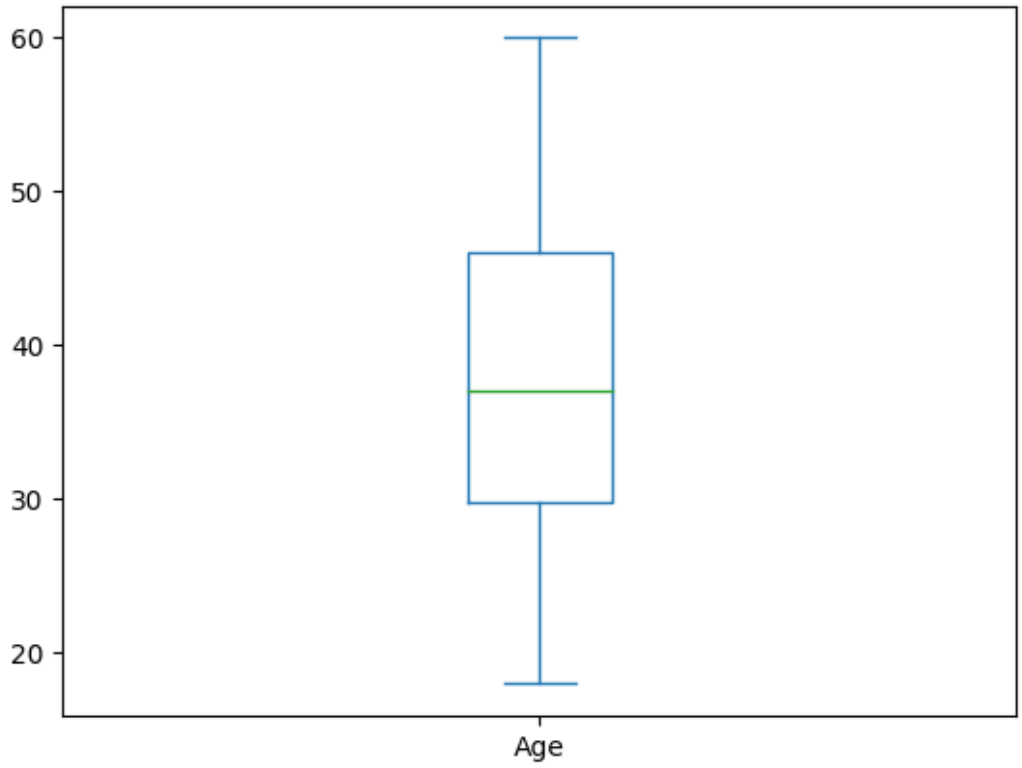
[26]:
```
df.hist()
```

[26]:
```
array([[<Axes: title={'center': 'User ID'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'EstimatedSalary'}>,
        <Axes: title={'center': 'Purchased'}>]], dtype=object)
```



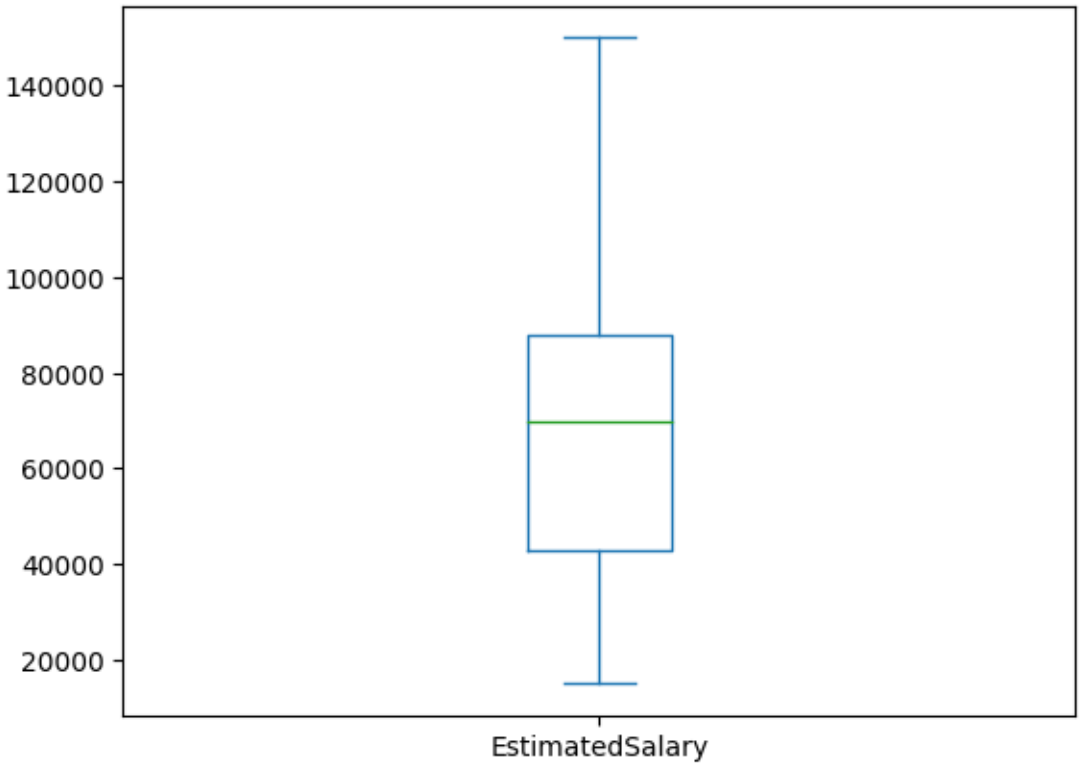
[27]:
```
df['Age'].plot.box()
```

[27]:
```
<Axes: >
```



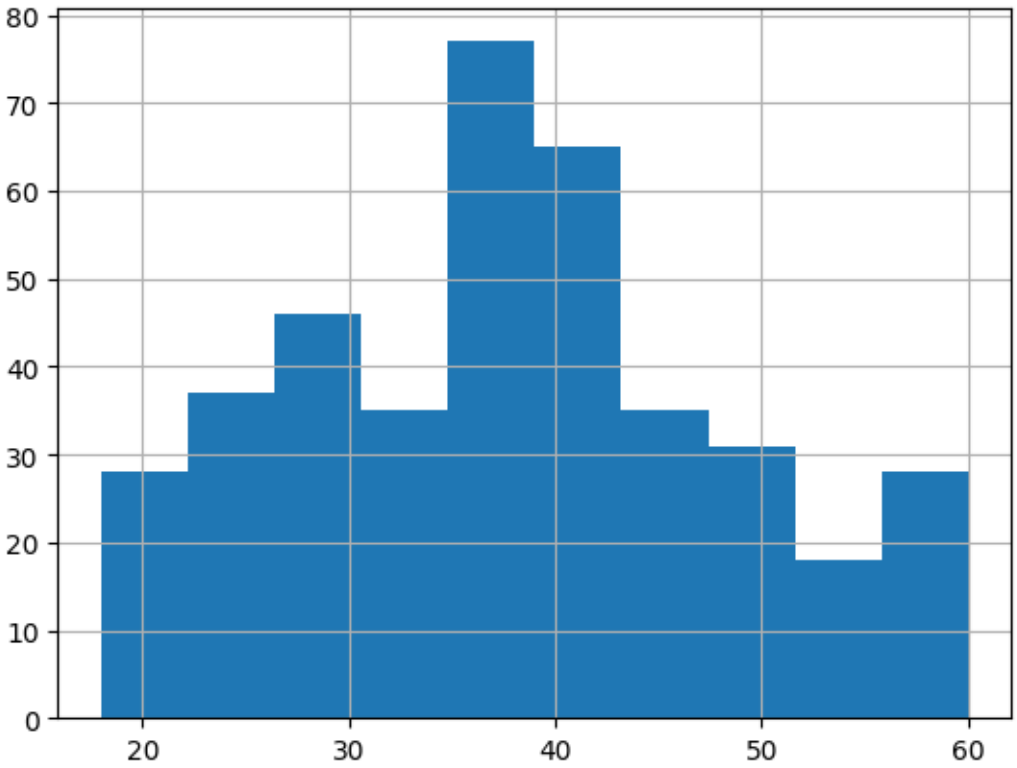
[28]:
```
df['EstimatedSalary'].plot.box()
```

[28]: <Axes: >



[29]:
```
df['Age'].hist()
```

[29]: <Axes: >



[30]:
```
X = df.drop('Purchased',axis=1)
```

[31]:
```
X
```

[31]:

	User ID	Gender	Age	EstimatedSalary
0	15624510	Male	19.0	19000.0
1	15810944	Male	35.0	20000.0
2	15668575	Female	26.0	43000.0
3	15603246	Female	27.0	57000.0
4	15804002	Male	19.0	76000.0
..	…	…	…	…
395	15691863	Female	46.0	41000.0
396	15706071	Male	51.0	23000.0
397	15654296	Female	50.0	20000.0
398	15755018	Male	36.0	33000.0
399	15594041	Female	49.0	36000.0

[400 rows x 4 columns]

[32]:
```
Y = df['Purchased']
```

[33]:
```
Y
```

[33]:
```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

[34]:
```
X.drop('Gender',inplace = True,axis=1)
```

[35]:
```
X
```

[35]:
```
       User ID   Age  EstimatedSalary
0     15624510  19.0          19000.0
1     15810944  35.0          20000.0
2     15668575  26.0          43000.0
3     15603246  27.0          57000.0
4     15804002  19.0          76000.0
..         …   …              …
395   15691863  46.0          41000.0
396   15706071  51.0          23000.0
397   15654296  50.0          20000.0
398   15755018  36.0          33000.0
399   15594041  49.0          36000.0

[400 rows x 3 columns]
```

[36]:
```
X.drop('User ID',inplace = True,axis=1)
```

[37]:
```
X
```

[37]:
```
       Age  EstimatedSalary
0     19.0          19000.0
1     35.0          20000.0
2     26.0          43000.0
3     27.0          57000.0
4     19.0          76000.0
..     …              …
395   46.0          41000.0
396   51.0          23000.0
397   50.0          20000.0
398   36.0          33000.0
```

```
399  49.0          36000.0

[400 rows x 2 columns]
```

[38]:
```
from sklearn.model_selection import train_test_split
```

[39]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.
↪25,random_state=0)
```

[40]:
```
x_test
```

[40]:
```
      Age  EstimatedSalary
132  30.0          87000.0
309  38.0          50000.0
341  35.0          75000.0
196  30.0          79000.0
246  35.0          50000.0
..    …              …
146  27.0          96000.0
135  23.0          63000.0
390  48.0          33000.0
264  48.0          90000.0
364  42.0         104000.0

[100 rows x 2 columns]
```

[41]:
```
x_train
```

[41]:
```
      Age  EstimatedSalary
250  44.0          39000.0
63   32.0         120000.0
312  38.0          50000.0
159  32.0         135000.0
283  52.0          21000.0
..    …              …
323  48.0          30000.0
192  29.0          43000.0
117  36.0          52000.0
47   27.0          54000.0
172  26.0         118000.0

[300 rows x 2 columns]
```

[42]:
```
y_test
```

[42]:
```
132    0
309    0
```

```
341    0
196    0
246    0
      ..
146    1
135    0
390    1
264    1
364    1
Name: Purchased, Length: 100, dtype: int64
```

[43]:
```
y_test
```

[43]:
```
132    0
309    0
341    0
196    0
246    0
      ..
146    1
135    0
390    1
264    1
364    1
Name: Purchased, Length: 100, dtype: int64
```

[44]:
```
y_train
```

[44]:
```
250    0
63     1
312    0
159    1
283    1
      ..
323    1
192    0
117    0
47     0
172    0
Name: Purchased, Length: 300, dtype: int64
```

[45]:
```
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
```

[46]:
```
x_train = std.fit_transform(x_train)
x_test = std.fit_transform(x_test)
```

[47]:
```
from sklearn.linear_model import LogisticRegression
```

[48]:
```
model = LogisticRegression()
```

[49]:
```
model.fit(x_train,y_train)
```

[49]:
```
LogisticRegression()
```

[50]:
```
y_pred = model.predict(x_test)
```

[51]:
```
from sklearn.metrics import␣
↪confusion_matrix,accuracy_score,precision_score,recall_score
CM = confusion_matrix(y_test,y_pred)
print(accuracy_score(y_test,y_pred))
```

```
0.87
```

[52]:
```
CM
```

[52]:
```
array([[63,  5],
       [ 8, 24]])
```

[52]:
```

```

[53]:
```
print(precision_score(y_test,y_pred))
```

```
0.8275862068965517
```

[54]:
```
print(recall_score(y_test,y_pred))
```

```
0.75
```

[55]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91        68
           1       0.83      0.75      0.79        32

    accuracy                           0.87       100
   macro avg       0.86      0.84      0.85       100
weighted avg       0.87      0.87      0.87       100
```

[56]:
```
myData = [[45,85000],[35,65000],[35,120000]]
```

[57]:
```
myData
```

[57]:
```
[[45, 85000], [35, 65000], [35, 120000]]
```

[58]:
```
myData = std.fit_transform(myData)
```

[59]:
```
myData
```

[59]:
```
array([[ 1.41421356, -0.21997067],
       [-0.70710678, -1.09985336],
       [-0.70710678,  1.31982404]])
```

[60]:
```
dataPred = model.predict(myData)
```

[61]:
```
dataPred
```

[61]:
```
array([1, 0, 0])
```

[62]:
```
#using min max scaling
from sklearn.preprocessing import MinMaxScaler
min_max_scale = MinMaxScaler()
```

[63]:
```
x_train = min_max_scale.fit_transform(x_train)
x_test = min_max_scale.fit_transform(x_test)
```

[64]:
```
model1= LogisticRegression()
```

[65]:
```
model1.fit(x_train,y_train)
```

[65]:
```
LogisticRegression()
```

[66]:
```
y_pred = model.predict(x_test)
```

[67]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.49      0.65        68
           1       0.48      1.00      0.65        32

    accuracy                           0.65       100
   macro avg       0.74      0.74      0.65       100
weighted avg       0.83      0.65      0.65       100
```

[67]:
```
```