In [ ]:
```
#import libraries
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from math import exp
```

In [ ]:
```
#load dataset
```

In [2]:
```
data = pd.read_csv("C:/Users/avcoe/Downloads/Social_Network_Ads.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	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

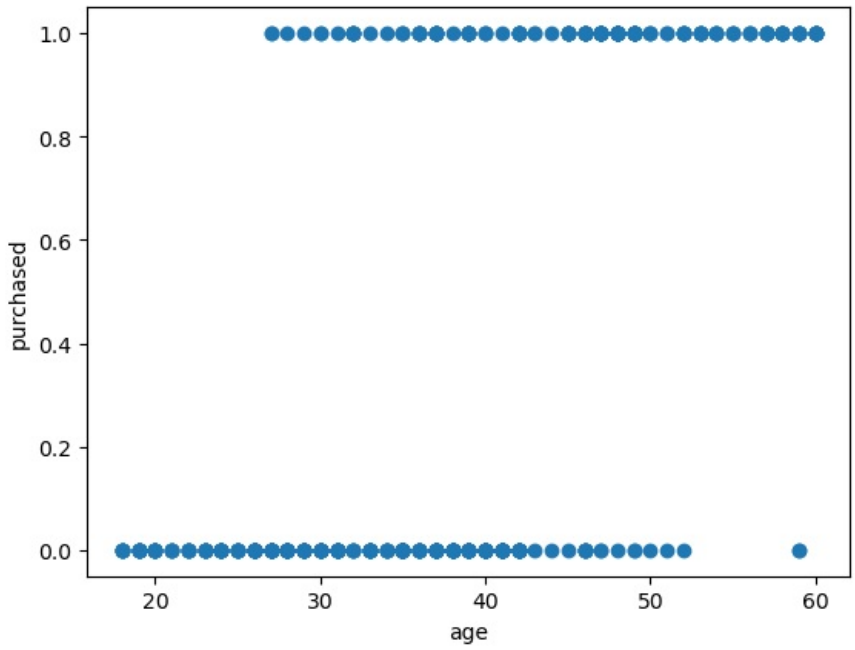
In [4]:
```
data.describe()
```

Out[4]:

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

In [ ]:
```
#visulizing the dataset
```

In [5]:
```
plt.scatter(data['Age'], data['Purchased'])
plt.xlabel("age")
plt.ylabel("purchased")
plt.show()
```



In [ ]:
```
# Divide the data to training set and test set
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(data['Age'], data['Purchased'],test_size=0.20)
```

In [ ]:
```
#making prediction usingscikit learn
```

In [7]:
```
from sklearn.linear_model import LogisticRegression
```

In [8]:
```
#create an instance and fit the model
```

In [9]:
```
model = LogisticRegression()
```

In [10]:
```
model.fit(X_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1).ravel())
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression()
```

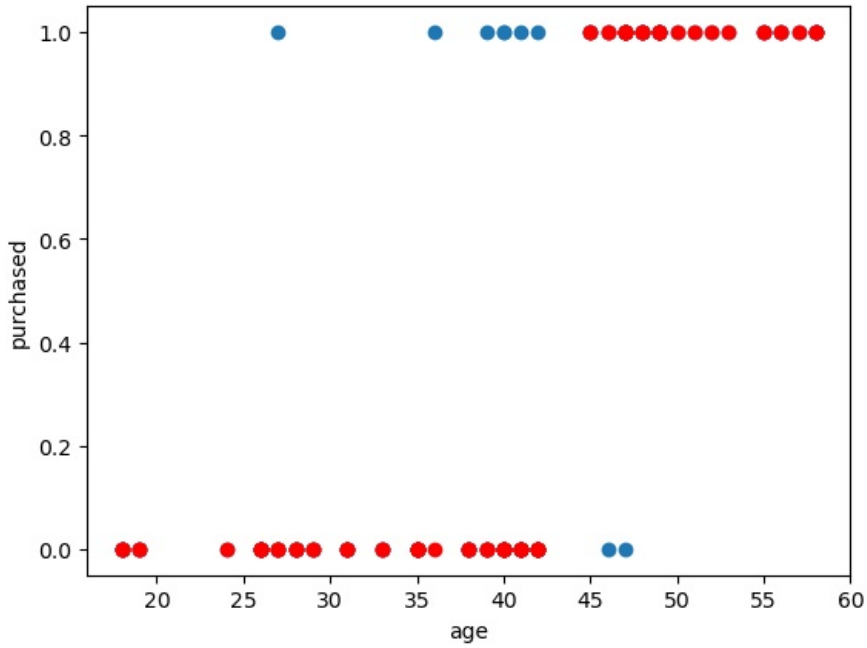
In [ ]:
```
#Making Predictions
```

In [11]:
```
y_pred_sk = model.predict(X_test.values.reshape(-1, 1))
plt.clf()
plt.scatter(x_test,y_test)
plt.scatter(X_test, y_pred_sk, c="red")
plt.xlabel("age")
plt.ylabel("purchased")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 3
      1 y_pred_sk = model.predict(X_test.values.reshape(-1, 1))
      2 plt.clf()
----> 3 plt.scatter(x_test,y_test)
      4 plt.scatter(X_test, y_pred_sk, c="red")
      5 plt.xlabel("age")

NameError: name 'x_test' is not defined
<Figure size 640x480 with 0 Axes>
```

In [17]:
```
y_pred_sk = model.predict(X_test.values.reshape(-1, 1))
plt.clf()
plt.scatter(X_test,y_test)
plt.scatter(X_test, y_pred_sk, c="red")
plt.xlabel("age")
plt.ylabel("purchased")
plt.show()
#Accuracy
print("Accuracy = {lr_model.score(X_test.values.reshape(-1, 1), y_test.values.reshape(-1, 1))}")
```




```
Accuracy = {lr_model.score(X_test.values.reshape(-1, 1), y_test.values.reshape(-1, 1))}
```

In [ ]:
```
```

In [18]:
```
from sklearn.metrics import confusion_matrix
#extracting true_positives, false_positives, true_negatives, false_negatives
tn, fp, fn, tp = confusion_matrix(y_test, y_pred_sk).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  45
False Positives:  2
False Negatives:  7
True Positives:  26
```

In [19]:
```
#Accuracy
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy {:0.2f}%:".format(Accuracy)
```

```
  Cell In[19], line 3
    print("Accuracy {:0.2f}%:".format(Accuracy)
                                               ^
SyntaxError: incomplete input
```

In [20]:
```
#Accuracy
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy {:0.2f}%:".format(Accuracy))
```

```
Accuracy 88.75%:
```

In [21]:
```
#Precision
Precision = tp/(tp+fp)
print("Precision {:0.2f}".format(Precision))
```

```
Precision 0.93
```

In [22]:
```
#Recall
Recall = tp/(tp+fn)
print("Recall {:0.2f}".format(Recall))
```

```
Recall 0.79
```

In [23]:
```
#Error rate
err = (fp + fn)/(tp + tn + fn + fp)
print("Error rate {:0.2f}".format(err))
```

```
Error rate 0.11
```

In [ ]: