

# untitled-1

February 28, 2024

```
[57]: import pandas as pd
import numpy as np
import seaborn as sns
```

```
[58]: df = pd.read_csv('Social_Network_Ads.csv')
df
```

```
[58]:
```

	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]

```
[59]: numerical_feat = [feat for feat in df.columns if df[feat].dtypes != 'O' and
↳ feat not in ['User ID']]
numerical_feat
```

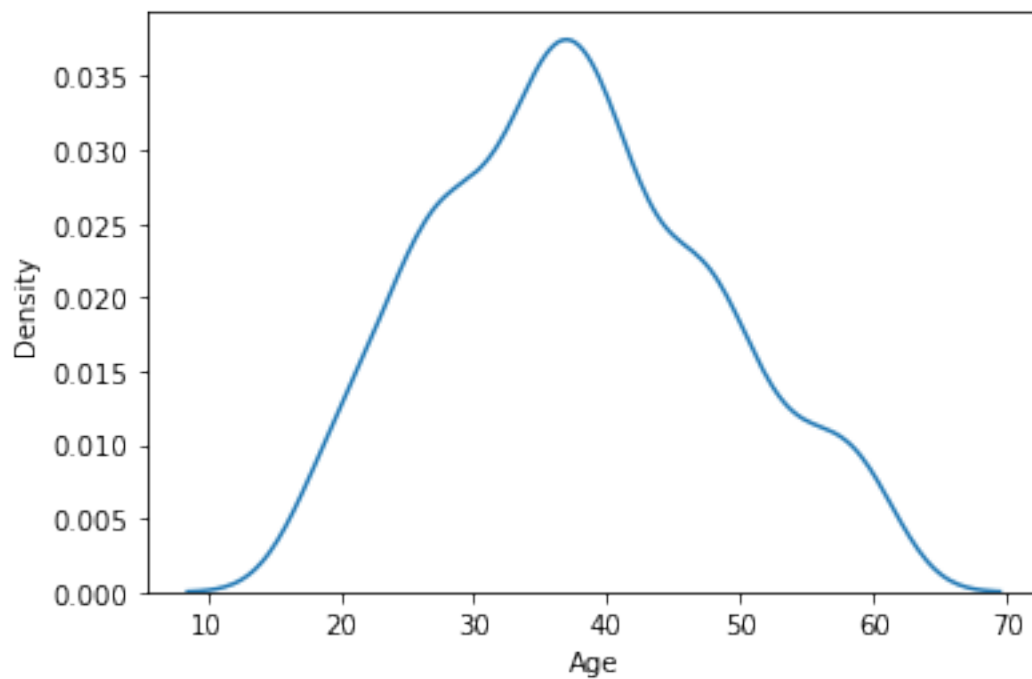
```
[59]: ['Age', 'EstimatedSalary', 'Purchased']
```

```
[60]: df[numerical_feat].isnull().sum()
```

```
[60]: Age          0
EstimatedSalary  0
Purchased       0
dtype: int64
```

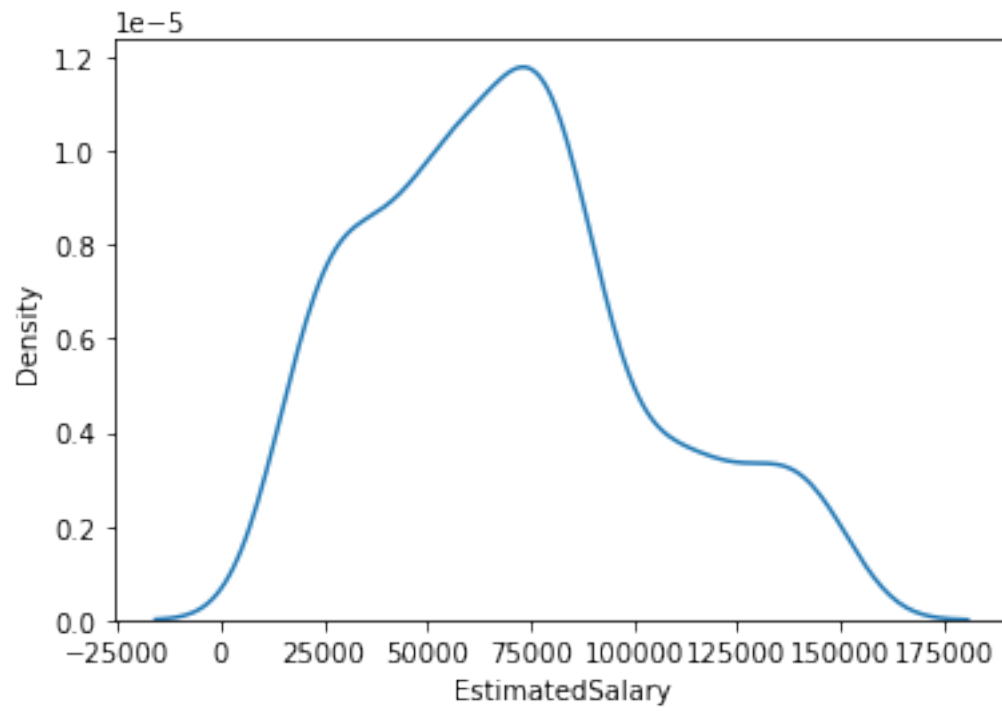
```
[61]: sns.kdeplot(data=df, x='Age')
```

```
[61]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```



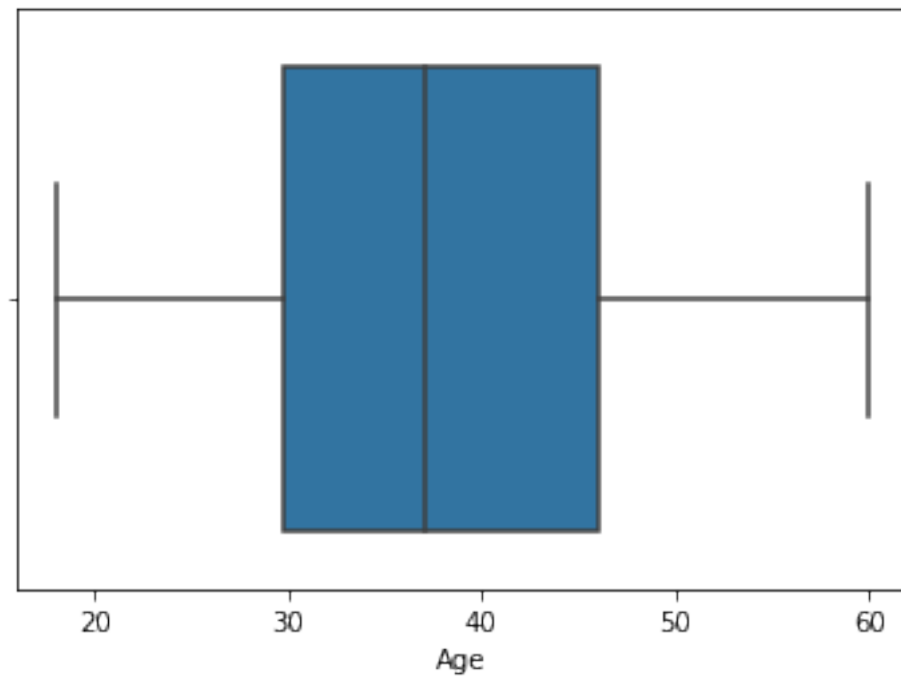
```
[62]: sns.kdeplot(data=df, x='EstimatedSalary')
```

```
[62]: <AxesSubplot:xlabel='EstimatedSalary', ylabel='Density'>
```



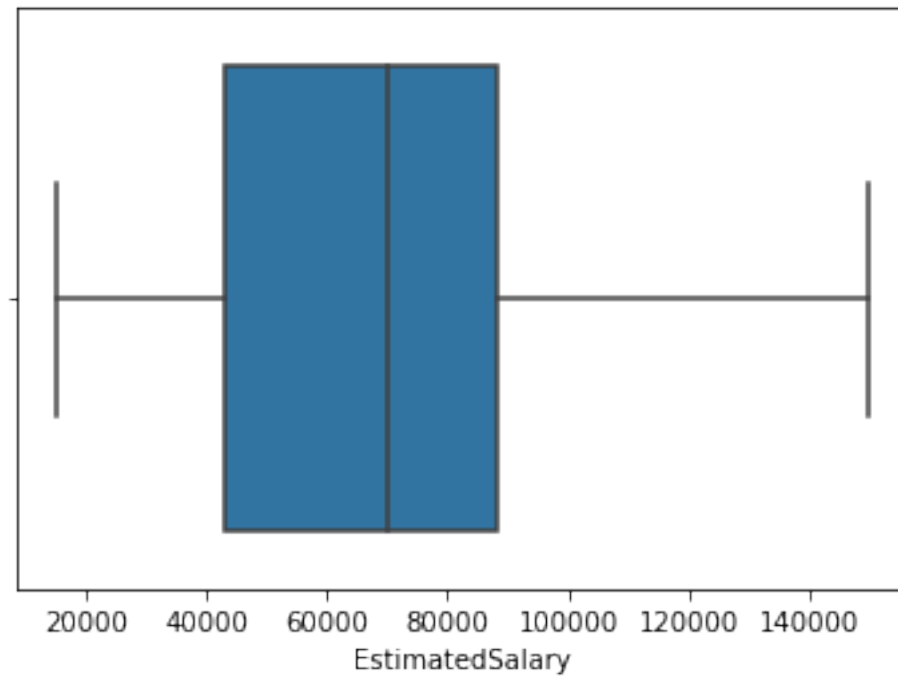
```
[63]: sns.boxplot(data=df,x='Age')
```

```
[63]: <AxesSubplot:xlabel='Age'>
```



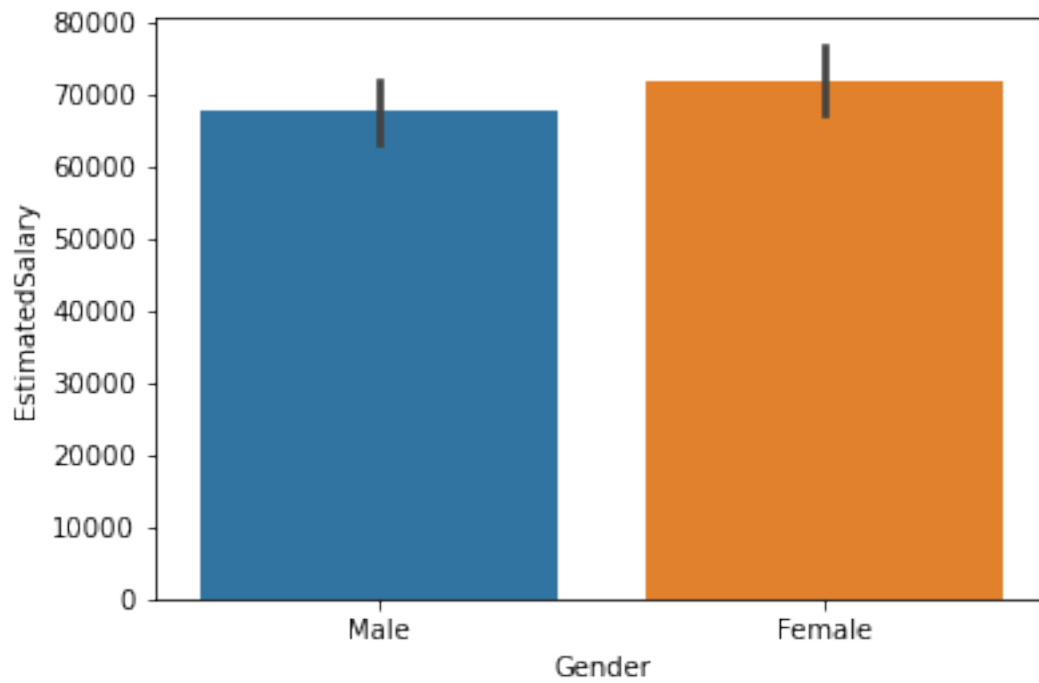
```
[64]: sns.boxplot(data=df,x='EstimatedSalary')
```

```
[64]: <AxesSubplot:xlabel='EstimatedSalary'>
```



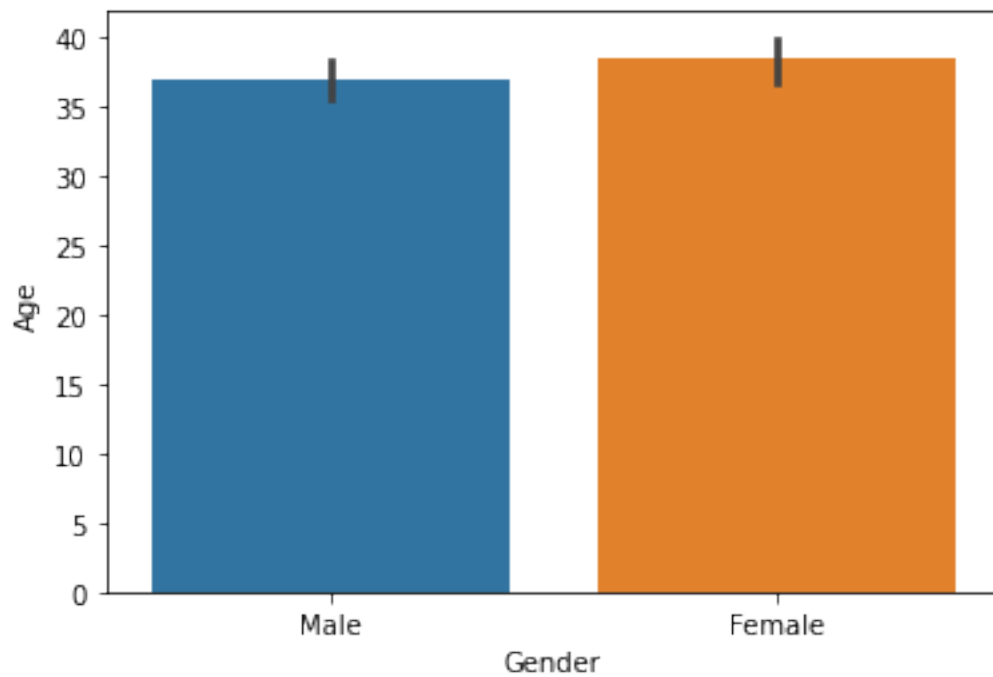
```
[65]: sns.barplot(data=df,x='Gender',y='EstimatedSalary')
```

```
[65]: <AxesSubplot:xlabel='Gender', ylabel='EstimatedSalary'>
```



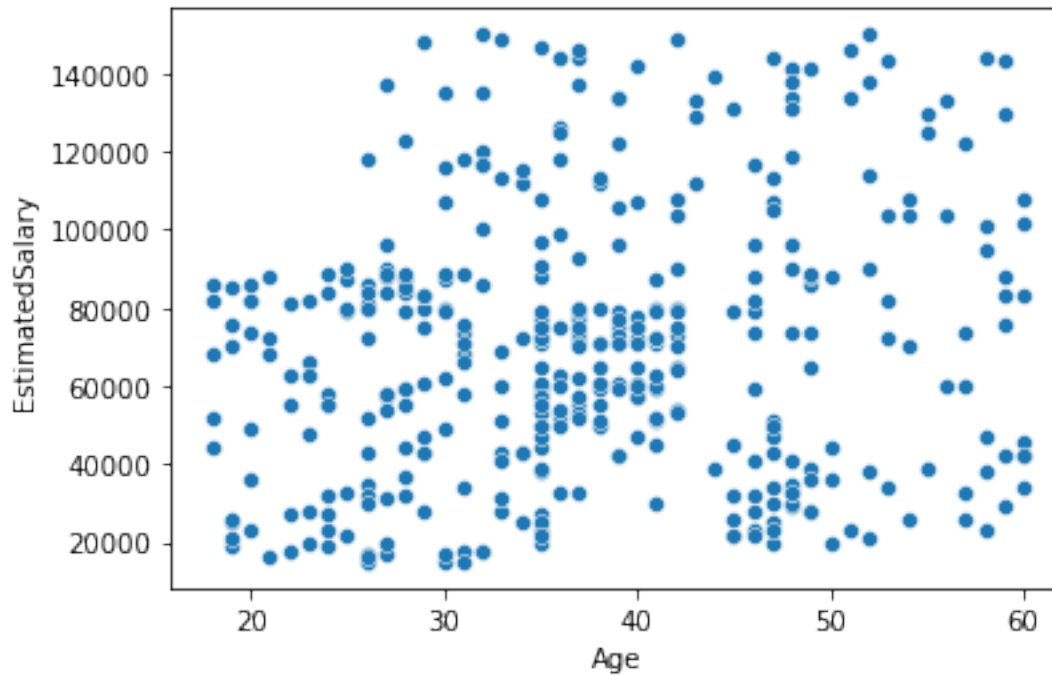
```
[66]: sns.barplot(data=df, x='Gender', y='Age')
```

```
[66]: <AxesSubplot:xlabel='Gender', ylabel='Age'>
```



```
[67]: sns.scatterplot(data=df, x='Age',y='EstimatedSalary')
```

```
[67]: <AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>
```



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

```
[69]: from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(df.
↳drop('Purchased',axis=1),df['Purchased'],test_size=0.3,random_state=0)
```

```
[70]: X_train
```

```
[70]:
```

	Gender	Age	EstimatedSalary
92	Male	26	15000
223	Male	60	102000
234	Female	38	112000
232	Male	40	107000
377	Female	42	53000
..	...	...	...
323	Female	48	30000
192	Male	29	43000
117	Male	36	52000
47	Female	27	54000

```
172  Female    26          118000
```

```
[280 rows x 3 columns]
```

```
[71]: Y_train
```

```
[71]: 92      0
      223    1
      234    0
      232    1
      377    0
      ..
      323    1
      192    0
      117    0
      47     0
      172    0
```

```
Name: Purchased, Length: 280, dtype: int64
```

```
[75]: #One hot encoding for gender column
      # from sklearn.preprocessing import OneHotEncoder
      # enc = OneHotEncoder()

      # enc_data = pd.DataFrame(enc.fit_transform(df[['Gender']]))

      # enc_data

      one_hot_encoded_data = pd.get_dummies(X_train, columns = ['Gender'])
      X_train_enc = pd.DataFrame(one_hot_encoded_data)
      X_train_enc
```

```
[75]:
```

	Age	EstimatedSalary	Gender_Female	Gender_Male
92	26	15000	0	1
223	60	102000	0	1
234	38	112000	1	0
232	40	107000	0	1
377	42	53000	1	0
..	...	...	...	...
323	48	30000	1	0
192	29	43000	0	1
117	36	52000	0	1
47	27	54000	1	0
172	26	118000	1	0

```
[280 rows x 4 columns]
```

```
[72]: one_hot_encoded_data_Y = pd.get_dummies(X_test, columns = ['Gender'])
X_test_enc = pd.DataFrame(one_hot_encoded_data_Y)
X_test_enc
```

```
[72]:
```

	Age	EstimatedSalary	Gender_Female	Gender_Male
132	30	87000	0	1
309	38	50000	1	0
341	35	75000	0	1
196	30	79000	1	0
246	35	50000	1	0
..	...	...	...	...
216	49	65000	0	1
259	45	131000	1	0
49	31	89000	1	0
238	46	82000	1	0
343	47	51000	1	0

[120 rows x 4 columns]

```
[76]: X_train_enc.drop('Gender_Male',axis=1,inplace=True)
```

```
[77]: X_train_enc.head()
```

```
[77]:
```

	Age	EstimatedSalary	Gender_Female
92	26	15000	0
223	60	102000	0
234	38	112000	1
232	40	107000	0
377	42	53000	1

```
[78]: X_test_enc.drop('Gender_Male',axis=1,inplace=True)
X_test_enc
```

```
[78]:
```

	Age	EstimatedSalary	Gender_Female
132	30	87000	0
309	38	50000	1
341	35	75000	0
196	30	79000	1
246	35	50000	1
..	...	...	...
216	49	65000	0
259	45	131000	1
49	31	89000	1
238	46	82000	1
343	47	51000	1

[120 rows x 3 columns]



```
[101]: # Standardization

# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# scaler.fit(X_train_enc)
# X_train_scaled = scaler.transform(X_train_enc)
# X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train_enc.columns)
# X_train_scaled
```

```
[101]:
```

	Age	EstimatedSalary	Gender_Female	Gender_Male
0	-1.163172	-1.584970	-0.992882	0.992882
1	2.170181	0.930987	-0.992882	0.992882
2	0.013305	1.220177	1.007169	-1.007169
3	0.209385	1.075582	-0.992882	0.992882
4	0.405465	-0.486047	1.007169	-1.007169
..	...	...	...	...
275	0.993704	-1.151185	1.007169	-1.007169
276	-0.869053	-0.775237	-0.992882	0.992882
277	-0.182774	-0.514966	-0.992882	0.992882
278	-1.065133	-0.457127	1.007169	-1.007169
279	-1.163172	1.393691	1.007169	-1.007169

[280 rows x 4 columns]

```
[79]: from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train_enc)
X_train_scaled = scaler.transform(X_train_enc)
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train_enc.columns)
X_train_scaled
```

```
[79]:
```

	Age	EstimatedSalary	Gender_Female
0	0.190476	0.000000	0.0
1	1.000000	0.644444	0.0
2	0.476190	0.718519	1.0
3	0.523810	0.681481	0.0
4	0.571429	0.281481	1.0
..	...	...	...
275	0.714286	0.111111	1.0
276	0.261905	0.207407	0.0
277	0.428571	0.274074	0.0
278	0.214286	0.288889	1.0
279	0.190476	0.762963	1.0

[280 rows x 3 columns]

```
[83]: X_test_scaled = scaler.transform(X_test_enc)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_train_enc.columns)
X_test_scaled
```

```
[83]:
```

	Age	EstimatedSalary	Gender_Female
0	0.285714	0.533333	0.0
1	0.476190	0.259259	1.0
2	0.404762	0.444444	0.0
3	0.285714	0.474074	1.0
4	0.404762	0.259259	1.0
..	...	...	...
115	0.738095	0.370370	0.0
116	0.642857	0.859259	1.0
117	0.309524	0.548148	1.0
118	0.666667	0.496296	1.0
119	0.690476	0.266667	1.0

[120 rows x 3 columns]

```
[85]: from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0).fit(X_train_scaled, Y_train)
```

```
[86]: predicted_values = clf.predict(X_test_scaled)
```

```
[87]: predicted_values.mean()
```

```
[87]: 0.225
```

```
[89]: from sklearn.metrics import (
    confusion_matrix,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score
)
```

```
[90]: conf_matrix = confusion_matrix(Y_test, predicted_values)
conf_matrix
```

```
[90]: array([[76,  3],
        [17, 24]])
```

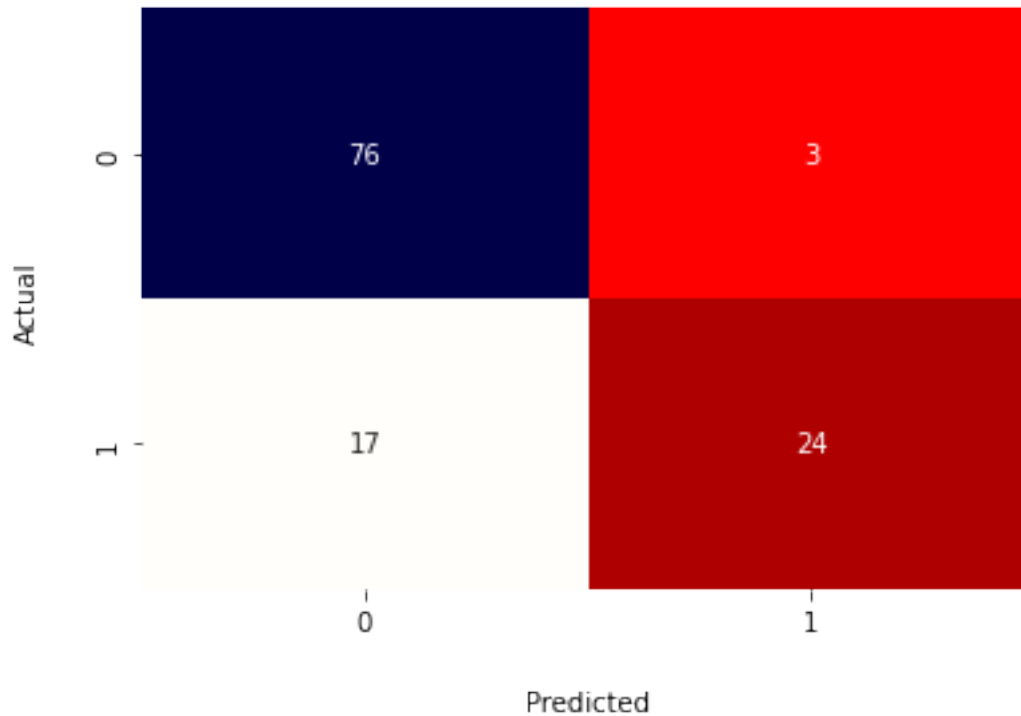
```
[93]: import matplotlib.pyplot as plt

ax = sns.heatmap(
    conf_matrix, # confusion matrix 2D array
    annot=True, # show numbers in the cells
```

```

    fmt='d', # show numbers as integers
    cbar=False, # don't show the color bar
    cmap='flag', # customize color map
    vmax=175 # to get better color contrast
)
ax.set_xlabel("Predicted", labelpad=20)
ax.set_ylabel("Actual", labelpad=20)
plt.show()

```



```

[94]: accuracy = accuracy_score(Y_test, predicted_values)
      print(f"Accuracy = {accuracy}")

```

Accuracy = 0.8333333333333334

```

[95]: precision = precision_score(Y_test, predicted_values)
      recall = recall_score(Y_test, predicted_values)
      f1score = f1_score(Y_test, predicted_values)

      print(f"Precision = {precision}")
      print(f"Recall = {recall}")
      print(f"F1 Score = {f1score}")

```

Precision = 0.8888888888888888

Recall = 0.5853658536585366

F1 Score = 0.7058823529411764

[ ]: