In [193…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [195…
```
dataset = pd.read_csv("Social_Network_Ads.csv")
dataset.head()
```

Out[195…

	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 [197…
```
dataset.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    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [199…
```
X = dataset.iloc[:, [2, 3]].values   # Age and Estimated Salary
y = dataset.iloc[:, 4].values        # Purchased
```

In [201…
```
#MODEL 1 : WITH random_state
```

In [202…
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0
)
```

In [204…
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [206…
```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[206…

▾ LogisticRegression ⓘ ?

▶ Parameters

In [208…
```
y_pred = classifier.predict(X_test)
```

```
y_pred
```

Out[208…
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1])
```

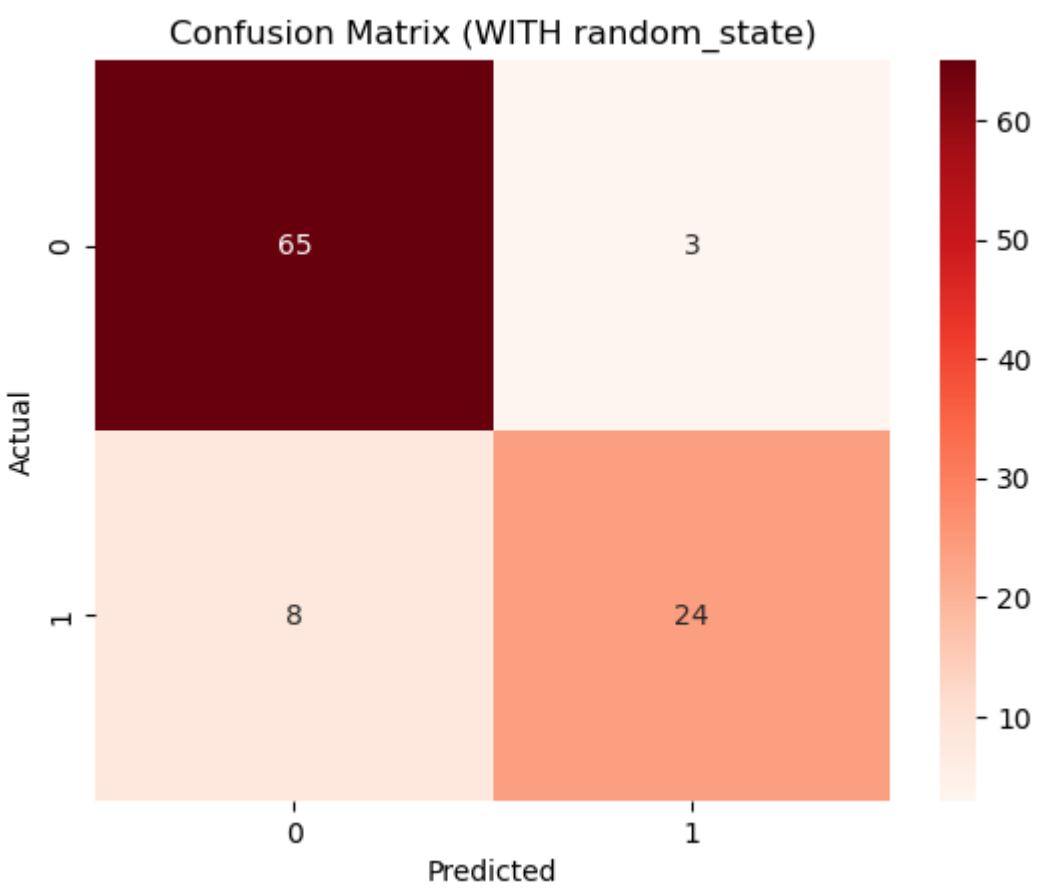
In [210…
```
from sklearn.metrics import confusion_matrix

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

Out[210…
```
array([[65,  3],
       [ 8, 24]])
```

In [212…
```
from sklearn.metrics import confusion_matrix

confusion_matrix_rs = confusion_matrix(y_test, y_pred)
print("Confusion Matrix (WITH random_state):\n", confusion_matrix_rs)

TN = confusion_matrix_rs[0][0]
FP = confusion_matrix_rs[0][1]
FN = confusion_matrix_rs[1][0]
TP = confusion_matrix_rs[1][1]

accuracy = (TP + TN) / (TP + TN + FP + FN)
error_rate = 1 - accuracy
precision = TP / (TP + FP)
recall = TP / (TP + FN)

print("\nWITH random_state")
print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
```

```
Confusion Matrix (WITH random_state):
 [[65  3]
 [ 8 24]]

WITH random_state
Accuracy: 0.89
Error Rate: 0.10999999999999999
Precision: 0.8888888888888888
Recall: 0.75
```

In [214…
```
sns.heatmap(confusion_matrix_rs, annot=True, fmt='d', cmap='Reds')
plt.title("Confusion Matrix (WITH random_state)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [216…

```
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train

X1, X2 = np.meshgrid(
    np.arange(X_set[:, 0].min() - 1, X_set[:, 0].max() + 1, 0.01),
    np.arange(X_set[:, 1].min() - 1, X_set[:, 1].max() + 1, 0.01)
)

plt.contourf(
    X1, X2,
    classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha=0.75,
    cmap=ListedColormap(('black', 'red'))
)

plt.scatter(
    X_set[:, 0], X_set[:, 1],
    c=y_set,
    cmap=ListedColormap(('black', 'red'))
)

plt.title("Logistic Regression (WITH random_state)")
plt.xlabel("Age")
plt.ylabel("Estimated Salary")
plt.show()
```

In [217…
```
#MODEL 2 : WITHOUT random_state
```

In [219…
```
X_train2, X_test2, y_train2, y_test2 = train_test_split(
    X, y, test_size=0.25
)
```

In [220…
```
scaler2 = StandardScaler()
X_train2 = scaler2.fit_transform(X_train2)
X_test2 = scaler2.transform(X_test2)
```

In [223…
```
classifier2 = LogisticRegression()
classifier2.fit(X_train2, y_train2)
```

Out[223…
▾ LogisticRegression ⓘ ⓘ

▶ Parameters

In [226…
```
y_pred2 = classifier2.predict(X_test2)
y_pred2
```

Out[226…
```
array([1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0])
```

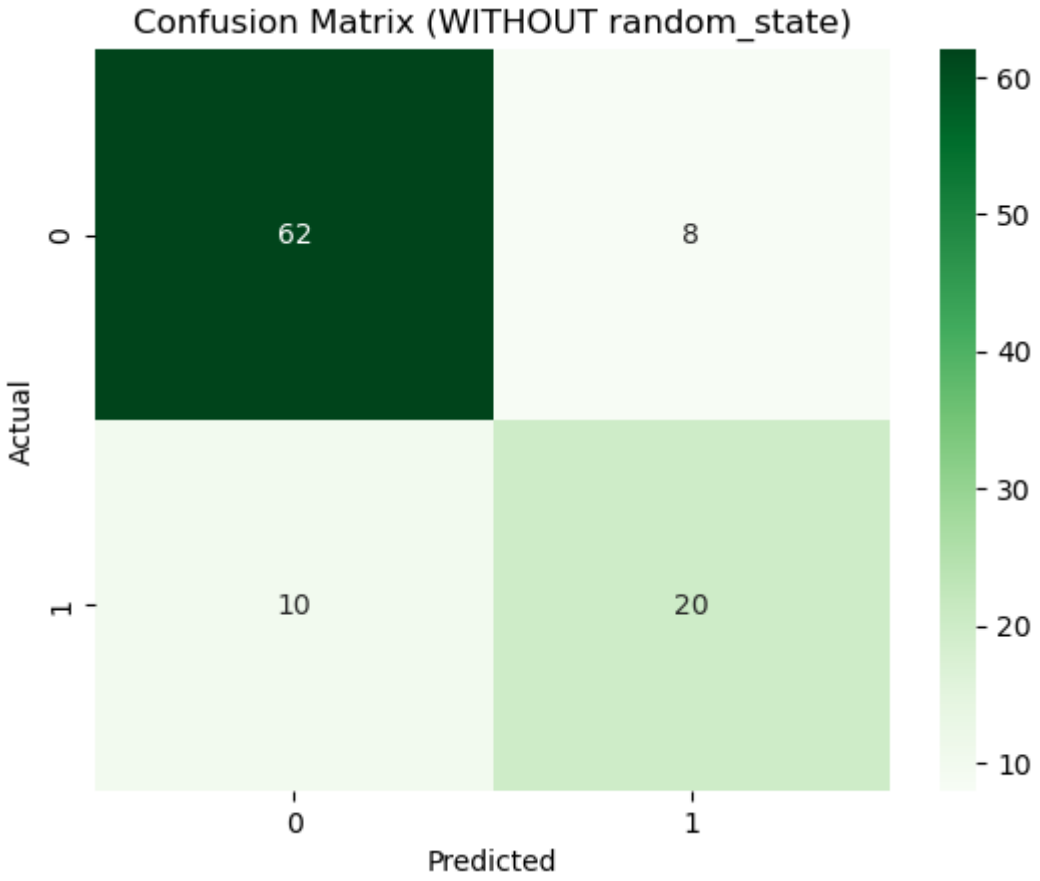
In [227…
```
confusion_matrix_no_rs = confusion_matrix(y_test2, y_pred2)
print("\nConfusion Matrix (WITHOUT random_state):\n", confusion_matrix_no_rs)

TN2 = confusion_matrix_no_rs[0][0]
FP2 = confusion_matrix_no_rs[0][1]
FN2 = confusion_matrix_no_rs[1][0]
TP2 = confusion_matrix_no_rs[1][1]

accuracy2 = (TP2 + TN2) / (TP2 + TN2 + FP2 + FN2)
```

```
error_rate2 = 1 - accuracy2
precision2 = TP2 / (TP2 + FP2)
recall2 = TP2 / (TP2 + FN2)

print("\nWITHOUT random_state")
print("Accuracy:", accuracy2)
print("Error Rate:", error_rate2)
print("Precision:", precision2)
print("Recall:", recall2)
```

```
Confusion Matrix (WITHOUT random_state):
 [[62  8]
 [10 20]]

WITHOUT random_state
Accuracy: 0.82
Error Rate: 0.18000000000000005
Precision: 0.7142857142857143
Recall: 0.6666666666666666
```

In [229…

```
sns.heatmap(confusion_matrix_no_rs, annot=True, fmt='d', cmap='Greens')
plt.title("Confusion Matrix (WITHOUT random_state)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [230…

```
X_set, y_set = X_train2, y_train2

X1, X2 = np.meshgrid(
    np.arange(X_set[:, 0].min() - 1, X_set[:, 0].max() + 1, 0.01),
    np.arange(X_set[:, 1].min() - 1, X_set[:, 1].max() + 1, 0.01)
)

plt.contourf(
    X1, X2,
    classifier2.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha=0.75,
    cmap=ListedColormap(('black', 'green'))
)
```

```
plt.scatter(
    X_set[:, 0], X_set[:, 1],
    c=y_set,
    cmap=ListedColormap(('black', 'green'))
)

plt.title("Logistic Regression (WITHOUT random_state)")
plt.xlabel("Age")
plt.ylabel("Estimated Salary")
plt.show()
```

In [ ]: