In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score, precision_score, recall_score, f1_score
```

In [2]:
```
data =  pd.read_csv('../input/Social_Network_Ads2.csv')
data.head(5)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipykernel_6456/1954719527.py in <module>
----> 1 data =  pd.read_csv('../input/Social_Network_Ads2.csv')
      2 data.head(5)

~/.local/lib/python3.10/site-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    209                 else:
    210                     kwargs[new_arg_name] = new_arg_value
--> 211             return func(*args, **kwargs)
    212
    213         return cast(F, wrapper)

~/.local/lib/python3.10/site-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

~/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na
_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, enc
oding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    948     kwds.update(kwds_defaults)
    949
--> 950     return _read(filepath_or_buffer, kwds)
    951
    952

~/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py in _read(filepath_or_buffer, kwds)
    603
    604     # Create the parser.
--> 605     parser = TextFileReader(filepath_or_buffer, **kwds)
    606
    607     if chunksize or iterator:

~/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py in __init__(self, f, engine, **kwds)
   1440
   1441         self.handles: IOHandles | None = None
-> 1442         self._engine = self._make_engine(f, self.engine)
   1443
   1444     def close(self) -> None:

~/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py in _make_engine(self, f, engine)
   1733                 if "b" not in mode:
   1734                     mode += "b"
-> 1735             self.handles = get_handle(
   1736                 f,
   1737                 mode,

~/.local/lib/python3.10/site-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    854         if ioargs.encoding and "b" not in ioargs.mode:
    855             # Encoding
--> 856             handle = open(
    857                 handle,
    858                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: '../input/Social_Network_Ads2.csv'
```

In [3]:
```
data =  pd.read_csv('Social_Network_Ads2.csv')
data.head(5)
```

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.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 [5]:
```
data.describe()
```

Out[5]:

	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 [6]:
```
X = dataset.iloc[:, [2, 3]].values
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_6456/3223724410.py in <module>
----> 1 X = dataset.iloc[:, [2, 3]].values

NameError: name 'dataset' is not defined
```

In [7]:
```
X = data.iloc[:, [2, 3]].values
```

In [8]:
```
y = data.iloc[:, 4].values

print(X[:3, :])
print('-'*15)
print(y[:3])

print('-'*15)
```

```
[[   19 19000]
 [   35 20000]
 [   26 43000]]
---------------
[0 0 0]
---------------
```

In [9]:
```
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 [10]:
```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])

print('-'*15)
```

```
[[   44  39000]
 [   32 120000]
 [   38  50000]]
---------------
[0 1 0]
---------------
[[   30 87000]
 [   38 50000]
 [   35 75000]]
---------------
[0 0 0]
---------------
```

In [11]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(300, 2)
(100, 2)
(300,)
(100,)
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [13]:
```
print(X_train[:3])
print('-'*15)
print(X_test[:3])

print('-'*15)
```

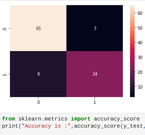
```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]]
---------------
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]]
---------------
```

In [14]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs' )
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(X_test[:10])
print('-'*15)
print(y_pred[:10])

print('-'*15)
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
---------------
[0 0 0 0 0 0 0 1 0 1]
---------------
```

In [15]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print("Confusion matrix : \n",cm)

Confusion matrix :
 [[65  3]
 [ 8 24]]
```

In [16]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cm,annot=True)
plt.show()
```



In [17]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is :",accuracy_score(y_test,y_pred)*100,'%')

Accuracy is : 90.0 %
```

```
  File "/tmp/ipykernel_6456/3263289604.py", line 4
    Accuracy is : 90.0 %
             ^
SyntaxError: invalid syntax
```

In [18]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is :",accuracy_score(y_test,y_pred)*100,'%')

Accuracy is : 89.0 %
```

In [19]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [20]:
```
precision=precision_score(y_test,y_pred)
print('Precision: %f' % precision)

Precision: 0.888889
```

In [21]:
```
recall=recall_score(y_test,y_pred)
print('Recall: %f' % recall)

Recall: 0.750000
```

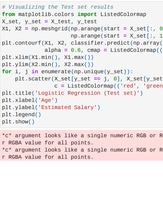
In [22]:
```
f1=f1_score(y_test,y_pred)
print('f1 score: %f' % f1)

F1 score: 0.813559
```

In [23]:
```
print(y_pred[:20])
print(y_test[:20])

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1]
```

In [24]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
```

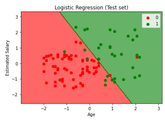
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB o
r RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB o
r RGBA value for all points.
```

Out[24]:
```
<matplotlib.legend.Legend at 0x7f114be92d70>
```



In [25]:
```
# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB o
r RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB o
r RGBA value for all points.
```


Logistic Regression (Test set)
Estimated Salary
Age
3
2
1
0
−1
−2
−2
−1
0
1
2
3

In [ ]: