Name : Tanishq Thuse

Div : SY CSAI - B

Roll no. : 60

PRN : 12310237

Subject : DV Assignment - 5

Assignment-5

Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset

Dataset link : https://www.kaggle.com/datasets/akram24/social-network-ads

Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Creating a dataframe & EDA

```
path = "/content/Social_Network_Ads.csv"
dataset = pd.read_csv(path)
```

```
dataset.head()
```

	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

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
dataset.tail()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0

```
dataset.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
		.655000	69742.500000	0.357500
		.482877	34096.960282	0.479864
		.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

Run cell (Ctrl+Enter)
cell executed since last change
executed by TANISHQ THUSE
9:35 PM (0 minutes ago)
executed in 1.034s

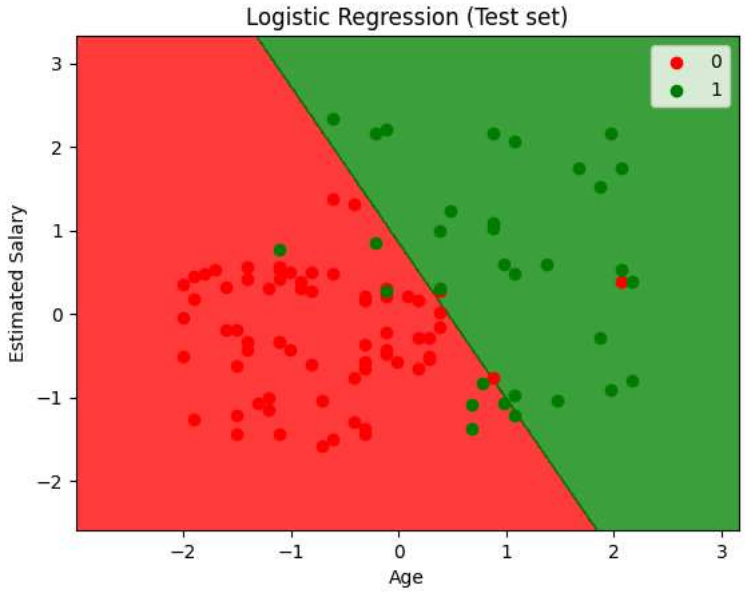
Logistic Regression

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values


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)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)


 # Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)


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, log_reg.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),1,
alpha = 0.75, 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()
plt.show()
```

```
<ipython-input-17-6cd9ba3d20df>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```


```
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, log_reg.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, 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()
```

```
<ipython-input-18-707d4e291116>:9: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```




```
# Calculating the accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", accuracy * 100)
```

```
Logistic Regression model accuracy (in %): 89.0
```