import numpy as np import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv("C:\\Users\\DELL\\Downloads\\Social\_Network\_Ads

(1).csv") df.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 4 15804002 Male 19 76000 0 df.shape (400, 5)

X = df.iloc[:, [2, 3]]

y = df.iloc[:,4].values

print(X) print(y)

Age EstimatedSalary

1. 19 19000
2. 35 20000
3. 26 43000
4. 27 57000
5. 19 76000 .. ... ... 395 46 41000
6. 51 23000
7. 50 20000
8. 36 33000
9. 49 36000

[400 rows x 2 columns]

[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0

0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0

0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
2. 0

0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0

1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 10 1
2. 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0
3. 1
4. 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0
5. 0
6. 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0

1 0

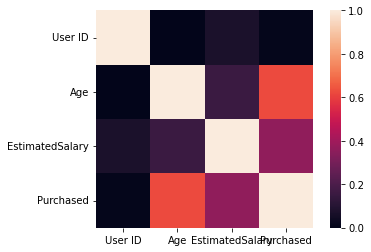
0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1

1. 1
2. 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]

import seaborn as sns correlations= df.corr()

sns.heatmap(correlations, square=True ) plt.yticks(rotation=0)

plt.xticks(rotation=0) plt.show()



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)

X\_train.shape

(300, 2)

X\_test.shape

(100, 2)

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train) X\_test = sc\_X.transform(X\_test)

from sklearn.linear\_model import LogisticRegression log\_reg = LogisticRegression(random\_state = 0) log\_reg.fit(X\_train, y\_train)

LogisticRegression(random\_state=0)

y\_pred = log\_reg.predict(X\_test)

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) print(cm)

[[65 3] [ 8 24]]

from sklearn.metrics import classification\_report cl\_report=classification\_report(y\_test,y\_pred) print(cl\_report)

precision recall f1-score support

0 0.89 0.96 0.92 68 1 0.89 0.75 0.81 32

accuracy 0.89 100 macro avg 0.89 0.85 0.87 100 weighted avg 0.89 0.89 0.89 100

*# Visualizing the Training set results* 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),

alpha = 0.6, cmap = ListedColormap(('red', 'green'))) plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

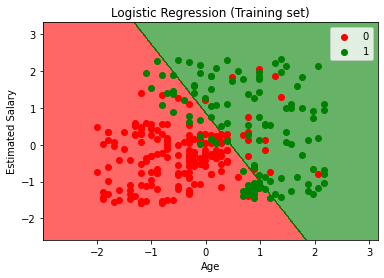
**for** i, j **in** enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Logistic Regression (Training set)')

plt.xlabel('Age') plt.ylabel('Estimated Salary') plt.legend()

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\* keywordargument or provide a 2D array with a single row if you intend to specify the same RGB or 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\* keywordargument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.



*# 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, log\_reg.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\* keywordargument or provide a 2D array with a single row if you intend to specify the same RGB or 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\* keywordargument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.

