Random Forest Classification

# Importing the libraries

### In [1] : fl

1mport numpy as np

import matplotlib.pyplot as plt import pandas as pd

**Importing the** dataset

In [2]: fl

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, :-1].values y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set



from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3,

In [3]: N

In [4]: N

X\_train.shape

Out[4]: (280, 2)

In [5]: N

y\_train.shape

Out[5]: (280,)

In [6]: 8

X\_test.shape

Out[6]: (120, 2)

In [7]: @

y\_test.shape

Out[7]: (120,)

# Feature Scaling

In [8]: N

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

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

In [9]: N

print(X\_train)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | [[-1.1631724 | -1.5849703 ] |  |
| [ 2.17018137 | 0.93098672] |
| [ 0.0133054 | 1.22017719] |  |
| [ 0.20938504 | 1.07558195] |  |
| [ 0.40546467 | *-8.* 48604654] |  |
| [-0.28081405 | -0.31253226] |  |
| [ 0.99370357 | -0.8330751 ] |  |
| [ 0.99370357 | 1.8563962 ] |  |
| [ 0.0133054 | 1.24909623] |  |
| [ -0.86905295 | 2.26126285] |  |
| [ -1. 1631724 | -1.5849703 ] |  |
| [ 2.17018137 | -0.80415605] |  |
| [-1.35925203 | -1.46929411] |  |
| [ 0.40546467 | 2.2901819 ] |  |
| [ 0.79762394 | 0.75747245] |  |
| [-0.96709276 | -0.31253226] |  |
| [ 0.11134522 | 0.75747245] |  |
| [-0.96709276 | 0.55503912] |  |
| [ 0.30742485 | 0.06341534] |  |
| In [10]: | 8 | print(X\_test) t'. X\*Jt'\*J\*WGW | -t\*. t\*WWWGt'\*JMj |  |
|  |  | [ 0.89566375 | 2.14558666] |  |
|  |  | [ 0.30742485 | -0.54388463] |  |
|  |  | [ 0.89566375 | 1.01774386] |  |
|  |  | [-1.45729185 | -1.2090227 ] |  |
|  |  | [ 1.09174339 | 2.05882953] |  |
|  |  | [-0.96709276 | 0.49720103] |  |
|  |  | [-0.86905295 | 0.29476771] |  |
|  |  | [-0.08473441 | -0.22577513] |  |
|  |  | [-0.5749335 | 0.46828198] |  |
|  |  | [-1.65337148 | 0.52612008] |  |
|  |  | [-0.08473441 | 0.26584866] |  |
|  |  | [ 1.87606192 | -0.28361322] |  |
|  |  | [-0.08473441 | -0.48604654] |  |
|  |  | [-1.35925203 | -0.34145131] |  |
|  |  | [-1.94749093 | -0.51496559] |  |
|  |  | [-1.55533166 | 0.32368675] |  |
|  |  | [-0.37885386 | -0.775237 ] |  |
| [-0.67297331 | | | -1.03550842] | |
| [ 1.09174339 | | | -0.97767033] | |

## Training the Random Forest Classification model on the Training set

In [11]: 8

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 20, criterion = 'entrop classifier.fit(X\_train, y\_train)

Out[11]: RandomForestClassifier(criterion='entropy', n\_estimators=20, random\_state

=0)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

## Predicting a new result

In [1]: 8

print(classifier.predict([[32,150000]]))

NameError Traceback (most recent call las

‹ipython-input-1-aa17d0668897> in <module>

----› 1 print(classifier.predict([[32,150000]]))

NameError: name 'classifier' is not defined

## Predicting the Test set results

In [13]: N

y\_pred = classifier.predict(X\_test)

*#print(np. concatenate((y\_pred.reshape(Len(y\_pred) 1) y\_test. reshape(Len(y*

## Making the Confusion Matrix

In [14]: N

\*\*om sklearn.metrics import confusion\_matrix, accuracy\_score cm = confusion\_matrix(y\_test, y\_pred)

print(cm) accuracy\_score(y\_test, y\_pred)

[ [74 5]

[ 5 36] ]

Out[14]: 0.9166666666666666

# Visualising the Training set results

In [15]: |d from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_sc np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_

plt.contourf(Xl, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 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)):

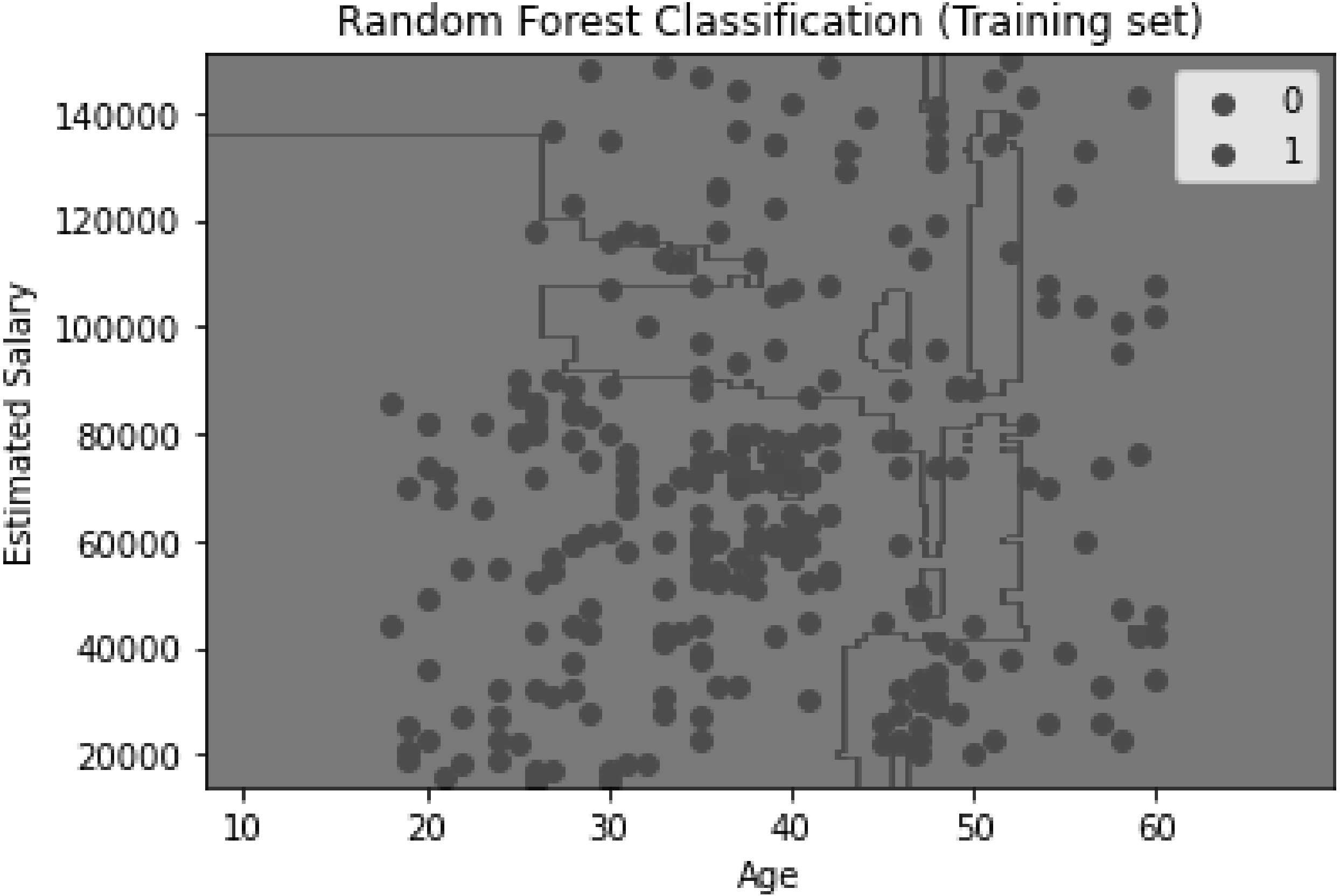
plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColc plt.title('Random Forest Classification (Training set)') plt.xlabel('Age')

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

plt.show()

\*c\* argument looks like a single numeric RGB or RGBA sequence, which shou ld be avoided as value-mapping will have precedence in case its length ma tches 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 or RGB A value for all points.

\*c\* argument looks like a single numeric RGB or RGBA sequence, which shou ld be avoided as value-mapping will have precedence in case its length ma tches 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 or RGB A value for all points.



# Visualising the Test set results

In [ ] : N

In [ ] : N

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_sc np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_

plt.contourf(Xl, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 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 = ListedColc plt.title('Random Forest Classification (Test set)')

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

plt.show()