**SOCIAL NETWORK Ads DATA**

The problem statement is well fitted using the Random Forest Classifier to get the desired output image. X and y represent the independent and dependent variables respectively. The dependent variable (y) is what we are trying to figure out and is influenced by the independent variables (X). In our case, the dependent variable is if the customer purchased the car or not. The independent variables are the age and estimated salary. “dataset.iloc[].values” is selecting the rows and columns we want from our dataset.

**Feature Scaling**

Feature scaling is a step that isn’t always needed when creating your machine learning model, but is in this case. If we put our numerical variables (i.e age, salary, etc.) in our classifier’s algorithm as they are, our results will become skewed. Even though age and salary represent two completely different things, the algorithm will plainly look at them as numbers. The purpose of feature scaling is to take every numerical value and put it on the same scale. This way the algorithm can use the values fairly. We use the StandardScaler module from the sklearn.preprocessing library to scale all of our independent variables.

After the data is standardized, the Random Forest Classifier is applied to the data for classification purpose. Because the task is to predict and classify whether a customer buys a car or not. Then to evaluate the results of the classifier, a confusion matrix and also the accuracy of the model is shown. Then visualizing the model by using the meshgrid function (used for changing 1-d array to a rectangular grid). A contourf graph type is used, because to fill the data points with the colours we expect. Then the scatter plot is used to plot the test data in the same graph and prove the prediction is right.

**GitHub Link:**

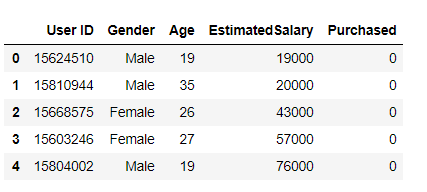
https://github.com/surya37-git/Social\_Network\_Ads

**CODE**

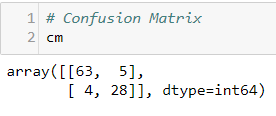
|  |
| --- |
|  |
|  |  |
|  | # Importing the libraries |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import pandas as pd |
|  |  |
|  | # Importing the dataset |
|  | dataset = pd.read\_csv('Customer\_Information.csv') |
|  | X = dataset.iloc[:, [2, 3]].values |
|  | y = dataset.iloc[:, 4].values |
|  |  |
|  | # Splitting the dataset into the Training set and Test set |
|  | from sklearn.cross\_validation 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) |
|  |  |
|  | # Fitting Random Forest to the Training set |
|  | from sklearn.ensemble import RandomForestClassifier |
|  | classifier = RandomForestClassifier(n\_estimators = 10, criterion = "entropy", random\_state = 0) |
|  | classifier.fit(X\_train, y\_train) |
|  |  |
|  | # Predicting the Test set results |
|  | y\_pred = classifier.predict(X\_test) |
|  |  |
|  | # Making the Confusion Matrix |
|  | from sklearn.metrics import confusion\_matrix |
|  | cm = confusion\_matrix(y\_test, y\_pred) |
|  |  |
|  | # Visualising 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, classifier.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('Random Forest (Training set)') |
|  | plt.xlabel('Age') |
|  | plt.ylabel('Estimated Salary') |
|  | plt.legend() |
|  | plt.show() |
|  |  |
|  | # Visualising 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 = 0.75, 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('Random Forest (Test set)') |
|  | plt.xlabel('Age') |
|  | plt.ylabel('Estimated Salary') |
|  | plt.legend() |
|  | plt.show() |

**OUTPUT**

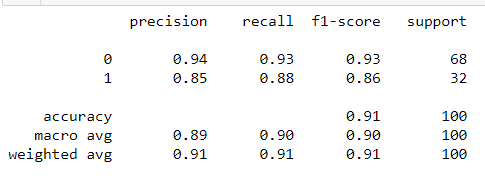
Data



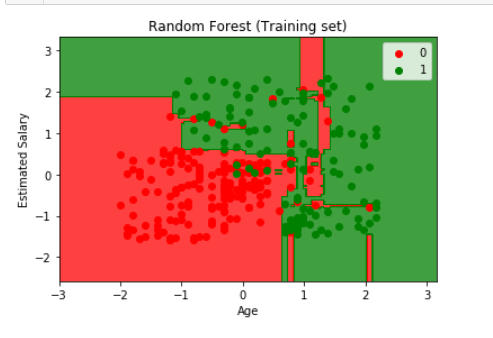
Confusion matrix



Accuracy



Plot(Training set)



Plot(Test set) – Desired output

