**DML Practicals**

**Practical 1**

**Simple Linear Regression**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Salary\_Data.csv')

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

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

print(X)

print(y)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

print(X\_train)

print(X\_test)

print(y\_train)

print(y\_test)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

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

plt.scatter(X\_train, y\_train, color = 'red')

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_test, regressor.predict(X\_test), color = 'blue')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

**Output:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Salary\_Data.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

print(X)

[[ 1.1]

[ 1.3]

[ 1.5]

[ 2. ]

[ 2.2]

[ 2.9]

[ 3. ]

[ 3.2]

[ 3.2]

[ 3.7]

[ 3.9]

[ 4. ]

[ 4. ]

[ 4.1]

[ 4.5]

[ 4.9]

[ 5.1]

[ 5.3]

[ 5.9]

[ 6. ]

[ 6.8]

[ 7.1]

[ 7.9]

[ 8.2]

[ 8.7]

[ 9. ]

[ 9.5]

[ 9.6]

[10.3]

[10.5]]

print(y)

[ 39343. 46205. 37731. 43525. 39891. 56642. 60150. 54445. 64445.

57189. 63218. 55794. 56957. 57081. 61111. 67938. 66029. 83088.

81363. 93940. 91738. 98273. 101302. 113812. 109431. 105582. 116969.

112635. 122391. 121872.]

print(X\_train)

[[ 2.9]

[ 5.1]

[ 3.2]

[ 4.5]

[ 8.2]

[ 6.8]

[ 1.3]

[10.5]

[ 3. ]

[ 2.2]

[ 5.9]

[ 6. ]

[ 3.7]

[ 3.2]

[ 9. ]

[ 2. ]

[ 1.1]

[ 7.1]

[ 4.9]

[ 4. ]]

print(X\_test)

[[ 1.5]

[10.3]

[ 4.1]

[ 3.9]

[ 9.5]

[ 8.7]

[ 9.6]

[ 4. ]

[ 5.3]

[ 7.9]]

print(y\_train)

[ 56642. 66029. 64445. 61111. 113812. 91738. 46205. 121872. 60150.

39891. 81363. 93940. 57189. 54445. 105582. 43525. 39343. 98273.

67938. 56957.]

print(y\_test)

[ 37731. 122391. 57081. 63218. 116969. 109431. 112635. 55794. 83088.

101302.]

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

LinearRegression()

y\_pred = regressor.predict(X\_test)

plt.scatter(X\_train, y\_train, color = 'red')

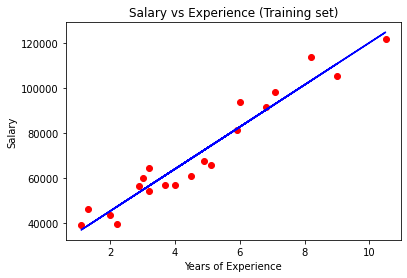
plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

****

plt.scatter(X\_test, y\_test, color = 'red')

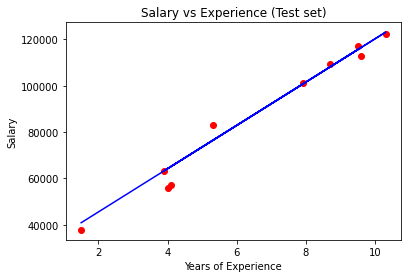
plt.plot(X\_test, regressor.predict(X\_test), color = 'blue')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()



**Practical 2**

**Multiple Linear Regression**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('50\_Startups.csv')

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

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

print(X)

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

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

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

**Output:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('50\_Startups.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

print(X)

[[165349.2 136897.8 471784.1 'New York']

[162597.7 151377.59 443898.53 'California']

[153441.51 101145.55 407934.54 'Florida']

[144372.41 118671.85 383199.62 'New York']

[142107.34 91391.77 366168.42 'Florida']

[131876.9 99814.71 362861.36 'New York']

[134615.46 147198.87 127716.82 'California']

[130298.13 145530.06 323876.68 'Florida']

[120542.52 148718.95 311613.29 'New York']

[123334.88 108679.17 304981.62 'California']

[101913.08 110594.11 229160.95 'Florida']

[100671.96 91790.61 249744.55 'California']

[93863.75 127320.38 249839.44 'Florida']

[91992.39 135495.07 252664.93 'California']

[119943.24 156547.42 256512.92 'Florida']

[114523.61 122616.84 261776.23 'New York']

[78013.11 121597.55 264346.06 'California']

[94657.16 145077.58 282574.31 'New York']

[91749.16 114175.79 294919.57 'Florida']

[86419.7 153514.11 0.0 'New York']

[76253.86 113867.3 298664.47 'California']

[78389.47 153773.43 299737.29 'New York']

[73994.56 122782.75 303319.26 'Florida']

[67532.53 105751.03 304768.73 'Florida']

[77044.01 99281.34 140574.81 'New York']

[64664.71 139553.16 137962.62 'California']

[75328.87 144135.98 134050.07 'Florida']

[72107.6 127864.55 353183.81 'New York']

[66051.52 182645.56 118148.2 'Florida']

[65605.48 153032.06 107138.38 'New York']

[61994.48 115641.28 91131.24 'Florida']

[61136.38 152701.92 88218.23 'New York']

[63408.86 129219.61 46085.25 'California']

[55493.95 103057.49 214634.81 'Florida']

[46426.07 157693.92 210797.67 'California']

[46014.02 85047.44 205517.64 'New York']

[28663.76 127056.21 201126.82 'Florida']

[44069.95 51283.14 197029.42 'California']

[20229.59 65947.93 185265.1 'New York']

[38558.51 82982.09 174999.3 'California']

[28754.33 118546.05 172795.67 'California']

[27892.92 84710.77 164470.71 'Florida']

[23640.93 96189.63 148001.11 'California']

[15505.73 127382.3 35534.17 'New York']

[22177.74 154806.14 28334.72 'California']

[1000.23 124153.04 1903.93 'New York']

[1315.46 115816.21 297114.46 'Florida']

[0.0 135426.92 0.0 'California']

[542.05 51743.15 0.0 'New York']

[0.0 116983.8 45173.06 'California']]

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

[[0.0 0.0 1.0 165349.2 136897.8 471784.1]

[1.0 0.0 0.0 162597.7 151377.59 443898.53]

[0.0 1.0 0.0 153441.51 101145.55 407934.54]

[0.0 0.0 1.0 144372.41 118671.85 383199.62]

[0.0 1.0 0.0 142107.34 91391.77 366168.42]

[0.0 0.0 1.0 131876.9 99814.71 362861.36]

[1.0 0.0 0.0 134615.46 147198.87 127716.82]

[0.0 1.0 0.0 130298.13 145530.06 323876.68]

[0.0 0.0 1.0 120542.52 148718.95 311613.29]

[1.0 0.0 0.0 123334.88 108679.17 304981.62]

[0.0 1.0 0.0 101913.08 110594.11 229160.95]

[1.0 0.0 0.0 100671.96 91790.61 249744.55]

[0.0 1.0 0.0 93863.75 127320.38 249839.44]

[1.0 0.0 0.0 91992.39 135495.07 252664.93]

[0.0 1.0 0.0 119943.24 156547.42 256512.92]

[0.0 0.0 1.0 114523.61 122616.84 261776.23]

[1.0 0.0 0.0 78013.11 121597.55 264346.06]

[0.0 0.0 1.0 94657.16 145077.58 282574.31]

[0.0 1.0 0.0 91749.16 114175.79 294919.57]

[0.0 0.0 1.0 86419.7 153514.11 0.0]

[1.0 0.0 0.0 76253.86 113867.3 298664.47]

[0.0 0.0 1.0 78389.47 153773.43 299737.29]

[0.0 1.0 0.0 73994.56 122782.75 303319.26]

[0.0 1.0 0.0 67532.53 105751.03 304768.73]

[0.0 0.0 1.0 77044.01 99281.34 140574.81]

[1.0 0.0 0.0 64664.71 139553.16 137962.62]

[0.0 1.0 0.0 75328.87 144135.98 134050.07]

[0.0 0.0 1.0 72107.6 127864.55 353183.81]

[0.0 1.0 0.0 66051.52 182645.56 118148.2]

[0.0 0.0 1.0 65605.48 153032.06 107138.38]

[0.0 1.0 0.0 61994.48 115641.28 91131.24]

[0.0 0.0 1.0 61136.38 152701.92 88218.23]

[1.0 0.0 0.0 63408.86 129219.61 46085.25]

[0.0 1.0 0.0 55493.95 103057.49 214634.81]

[1.0 0.0 0.0 46426.07 157693.92 210797.67]

[0.0 0.0 1.0 46014.02 85047.44 205517.64]

[0.0 1.0 0.0 28663.76 127056.21 201126.82]

[1.0 0.0 0.0 44069.95 51283.14 197029.42]

[0.0 0.0 1.0 20229.59 65947.93 185265.1]

[1.0 0.0 0.0 38558.51 82982.09 174999.3]

[1.0 0.0 0.0 28754.33 118546.05 172795.67]

[0.0 1.0 0.0 27892.92 84710.77 164470.71]

[1.0 0.0 0.0 23640.93 96189.63 148001.11]

[0.0 0.0 1.0 15505.73 127382.3 35534.17]

[1.0 0.0 0.0 22177.74 154806.14 28334.72]

[0.0 0.0 1.0 1000.23 124153.04 1903.93]

[0.0 1.0 0.0 1315.46 115816.21 297114.46]

[1.0 0.0 0.0 0.0 135426.92 0.0]

[0.0 0.0 1.0 542.05 51743.15 0.0]

[1.0 0.0 0.0 0.0 116983.8 45173.06]]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

y\_pred = regressor.predict(X\_test)

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

[[103015.2 103282.38]

[132582.28 144259.4 ]

[132447.74 146121.95]

[ 71976.1 77798.83]

[178537.48 191050.39]

[116161.24 105008.31]

[ 67851.69 81229.06]

[ 98791.73 97483.56]

[113969.44 110352.25]

[167921.07 166187.94]]

**Practical 3**

**K-Nearest Neighbour**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

y = dataset.iloc[:, -1].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)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

print(classifier.predict(sc.transform([[40,200000]])))

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

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

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\_set[:, 0].max() + 10, step = 1),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 1))

plt.contourf(X1, X2, classifier.predict(sc.transform(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('K-NN (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**Output:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

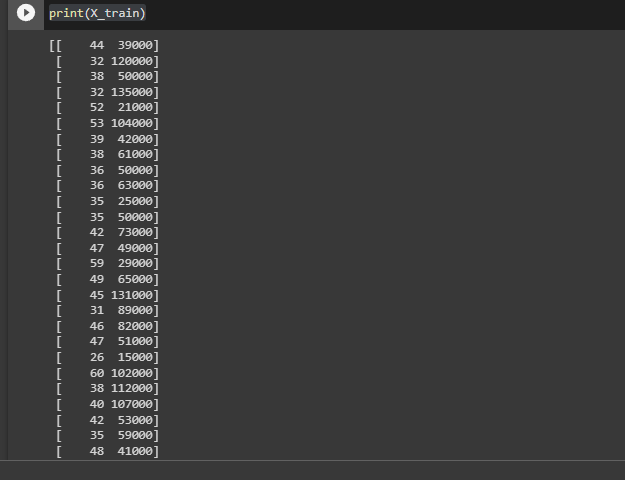
X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

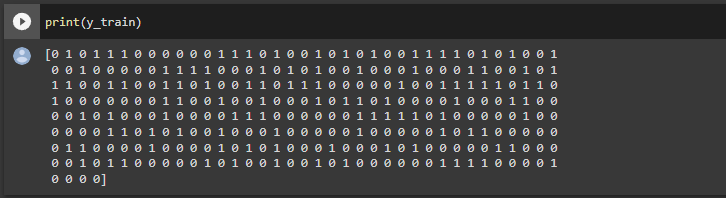
from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

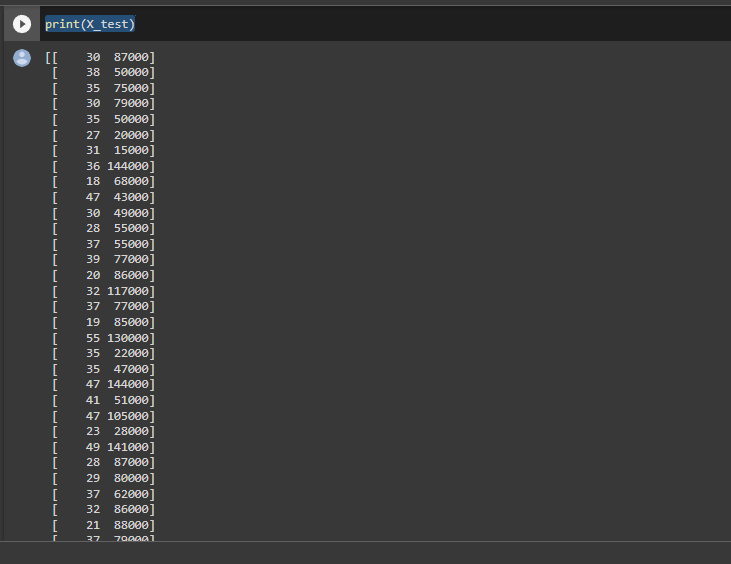
print(X\_train)



print(y\_train)



print(X\_test)



print(y\_test)

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

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

0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]

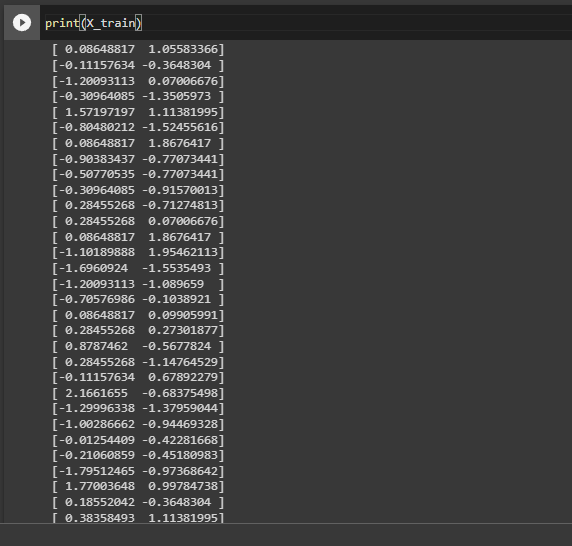
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

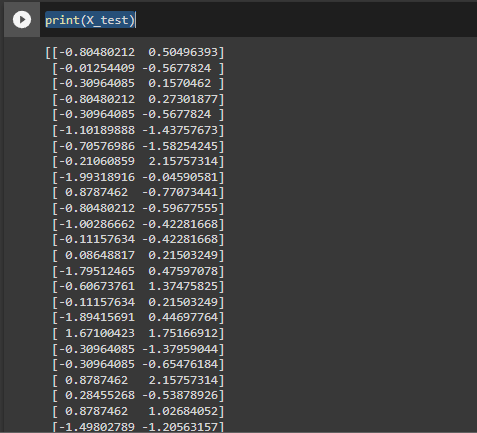
X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)



print(X\_test)



from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

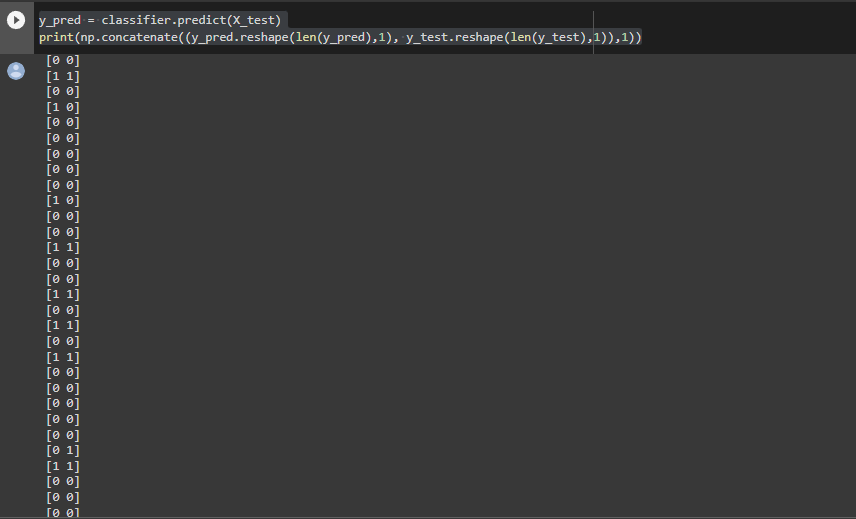
KNeighborsClassifier()

print(classifier.predict(sc.transform([[40,200000]])))

[1]

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))



from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

[[64 4]

[ 3 29]]

0.93

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\_set[:, 0].max() + 10, step = 1),

                     np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 1))

plt.contourf(X1, X2, classifier.predict(sc.transform(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)

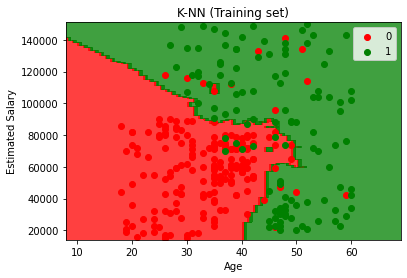
plt.title('K-NN (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



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\_set[:, 0].max() + 10, step = 1),

                     np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 1))

plt.contourf(X1, X2, classifier.predict(sc.transform(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)

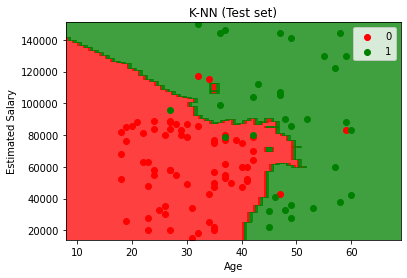
plt.title('K-NN (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



**Practical 4**

**Support Vector Machine**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

y = dataset.iloc[:, -1].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)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

print(classifier.predict(sc.transform([[30,87000]])))

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

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

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\_set[:, 0].max() + 10, step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(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('SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

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\_set[:, 0].max() + 10, step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(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('SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**Output:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

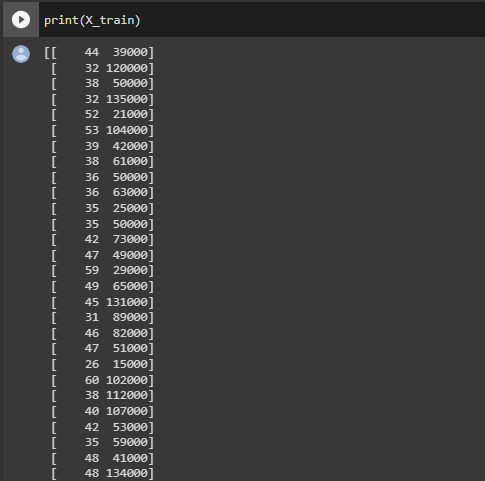
X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].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)

print(X\_train)



print(y\_train)

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

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

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

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

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

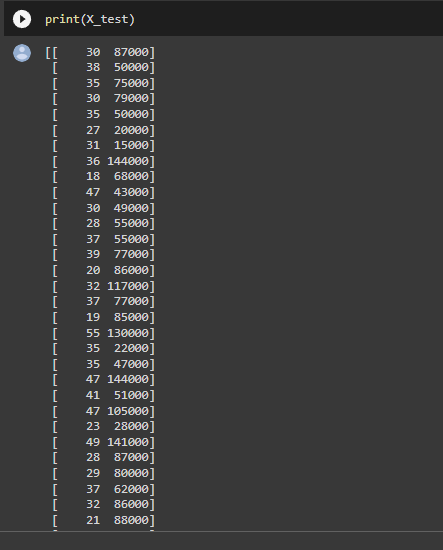
0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0

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

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

0 0 0 0]

print(X\_test)



print(y\_test)

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

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

0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]

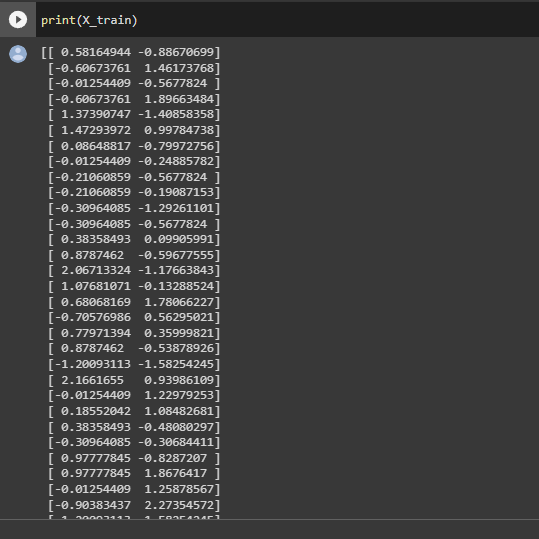
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

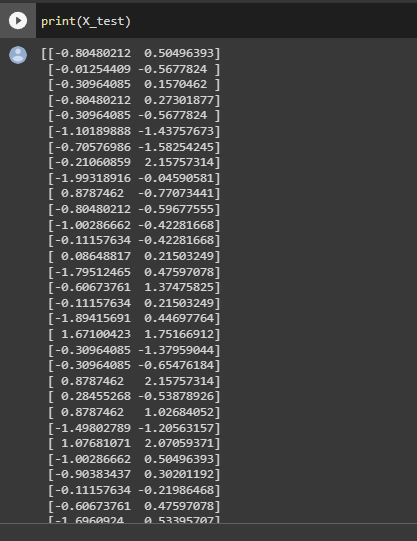
X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)



print(X\_test)



from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

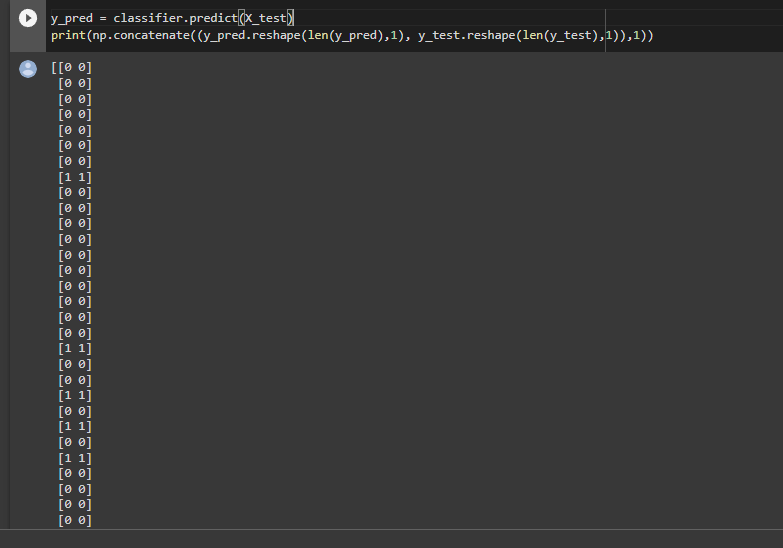
SVC(kernel='linear', random\_state=0)

print(classifier.predict(sc.transform([[30,87000]])))

[0]

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))



from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

[[66 2]

[ 8 24]]

0.9

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\_set[:, 0].max() + 10, step = 0.25),

                     np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(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)

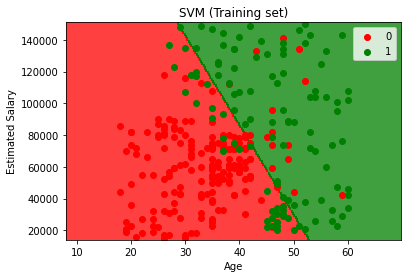
plt.title('SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



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\_set[:, 0].max() + 10, step = 0.25),

                     np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(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)

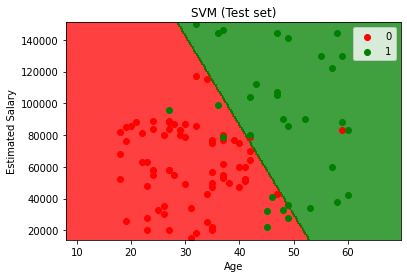
plt.title('SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



**Practical 5**

**K-Means Clustering**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

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

print(X)

from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

print(y\_kmeans)

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**Output:-**

import numpy as np

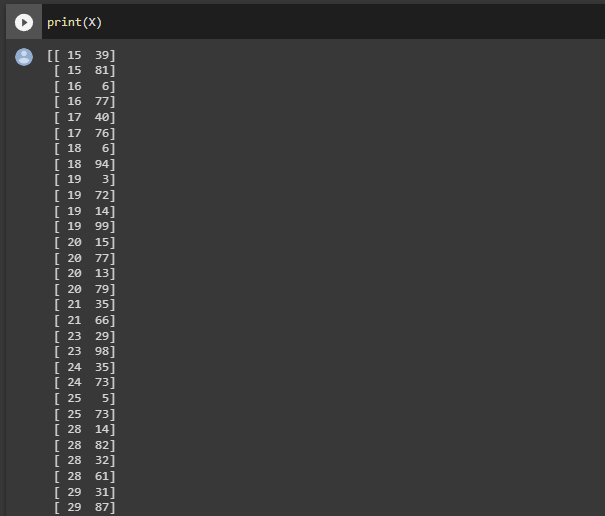
import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

print(X)



from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

    kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

    kmeans.fit(X)

    wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

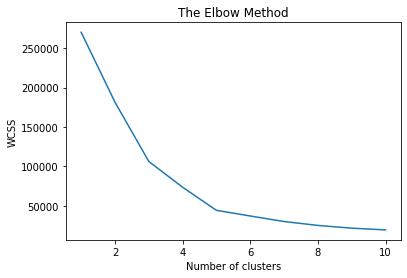
print(y\_kmeans)

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

print(y\_kmeans)



from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

print(y\_kmeans)

[2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2

3 2 3 2 3 2 0 2 3 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 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 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 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4

1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1

4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')

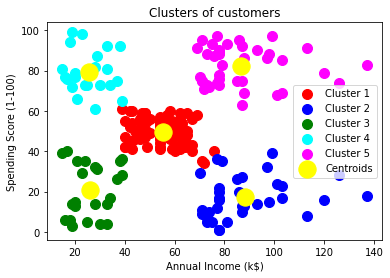
plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()



**Practical 6**

**Hierarchical Clustering**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

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

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**Practical 7**

**Build ANN**

**Code:-**

import numpy as np

import pandas as pd

import tensorflow as tf

tf.\_\_version\_\_

dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:, 3:-1].values

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

print(X)

print(y)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:, 2] = le.fit\_transform(X[:, 2])

print(X)

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

X\_test = sc.transform(X\_test)

ann = tf.keras.models.Sequential()

ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ann.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 100)

print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)

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

y\_pred = (y\_pred > 0.5)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

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

**Output:-**

import numpy as np

import pandas as pd

import tensorflow as tf

tf.\_\_version\_\_

'2.2.0'

dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:, 3:-1].values

y = dataset.iloc[:, -1].values

print(X)

[[619 'France' 'Female' ... 1 1 101348.88]

[608 'Spain' 'Female' ... 0 1 112542.58]

[502 'France' 'Female' ... 1 0 113931.57]

...

[709 'France' 'Female' ... 0 1 42085.58]

[772 'Germany' 'Male' ... 1 0 92888.52]

[792 'France' 'Female' ... 1 0 38190.78]]

print(y)

[1 0 1 ... 1 1 0]

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:, 2] = le.fit\_transform(X[:, 2])

print(X)

[[619 'France' 0 ... 1 1 101348.88]

[608 'Spain' 0 ... 0 1 112542.58]

[502 'France' 0 ... 1 0 113931.57]

...

[709 'France' 0 ... 0 1 42085.58]

[772 'Germany' 1 ... 1 0 92888.52]

[792 'France' 0 ... 1 0 38190.78]]

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

[[1.0 0.0 0.0 ... 1 1 101348.88]

[0.0 0.0 1.0 ... 0 1 112542.58]

[1.0 0.0 0.0 ... 1 0 113931.57]

...

[1.0 0.0 0.0 ... 0 1 42085.58]

[0.0 1.0 0.0 ... 1 0 92888.52]

[1.0 0.0 0.0 ... 1 0 38190.78]]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

X\_test = sc.transform(X\_test)

ann = tf.keras.models.Sequential()

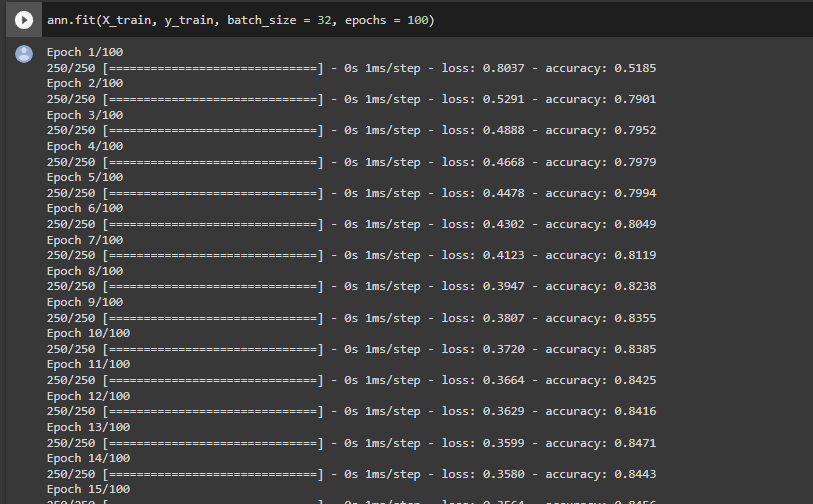
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ann.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 100)



print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)

[[False]]

y\_pred = ann.predict(X\_test)

y\_pred = (y\_pred > 0.5)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

[[0 0]

[0 1]

[0 0]

...

[0 0]

[0 0]

[0 0]]

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

[[1516 79]

[ 200 205]]

0.8605

**Practical 8**

**CNN**

**Code:-**

import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

tf.\_\_version\_\_

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

training\_set = train\_datagen.flow\_from\_directory('C:/Users/micha/Downloads/dataset/training\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

test\_datagen = ImageDataGenerator(rescale = 1./255)

test\_set = test\_datagen.flow\_from\_directory('C:/Users/micha/Downloads/dataset/test\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

cnn.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

cnn.fit(x = training\_set, validation\_data = test\_set, epochs = 25)

import numpy as np

from tensorflow.keras.preprocessing import image

test\_image = image.load\_img('C:/Users/micha/Downloads/dataset/single\_prediction/cat\_or\_dog\_2.jpg', target\_size = (64, 64))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = cnn.predict(test\_image)

training\_set.class\_indices

if result[0][0] == 1:

prediction = 'dog'

else:

prediction = 'cat'

print(prediction)

**Output:-**

import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

tf.\_\_version\_\_

'2.10.0'

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

training\_set = train\_datagen.flow\_from\_directory('C:/Users/micha/Downloads/dataset/training\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

Found 10 images belonging to 2 classes.

test\_datagen = ImageDataGenerator(rescale = 1./255)

test\_set = test\_datagen.flow\_from\_directory('C:/Users/micha/Downloads/dataset/test\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

Found 10 images belonging to 2 classes.

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

cnn.add(tf.keras.layers.Flatten())

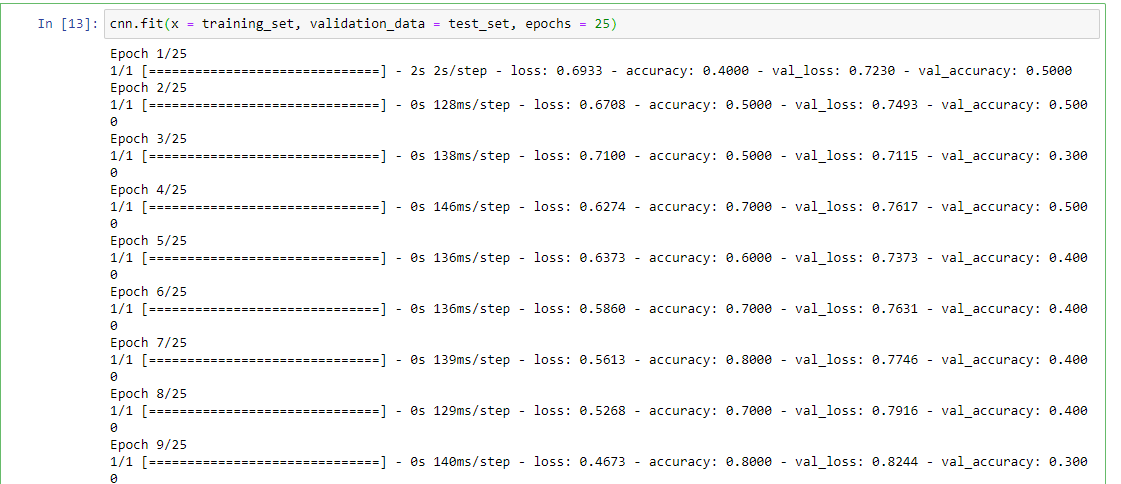
cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

cnn.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

cnn.fit(x = training\_set, validation\_data = test\_set, epochs = 25)



import numpy as np

from tensorflow.keras.preprocessing import image

test\_image = image.load\_img('C:/Users/micha/Downloads/dataset/single\_prediction/cat\_or\_dog\_2.jpg', target\_size = (64, 64))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = cnn.predict(test\_image)

training\_set.class\_indices

if result[0][0] == 1:

prediction = 'dog'

else:

prediction = 'cat'

1/1 [==============================] - 0s 109ms/step

In [19]:

print(prediction)

cat