Name: Hemil Mehul Shah

Email id: hxshah4@shockers.wichita.edu

Under-grads

Exp#1: Develop a system based on two-class **Support Vector Machine (SVM)** that can predict if the subject will purchase iPhone.

Dataset:
https://github.com/omairaasim/machine_learning/blob/master/project_11_k_nearest_neighbor/iphone_purchase_records.csv

Training/ Test Split: 75-25

Deliverables:
1. Code (50 points)
2. Report outlining the steps performed and the results obtained (50 points)

Code:
Code below is also attached as separate file to the homework.

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 78,
   "id": "0e8bbb4f",
   "metadata": {},
   "outputs": [],
   "source": [
    "#iphone.csv is the name of the file that is saved in the same directory as this code\n",
    "df = pd.read_csv(\"iphone.csv\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 79,
   "id": "fab33524",
   "metadata": {},
   "outputs": [
    {
     "data": {
```

```
   "text/plain": [
    "(400, 4)"
   ]
  },
  "execution_count": 79,
  "metadata": {},
  "output_type": "execute_result"
  }
 ],
 "source": [
  "#shape of the data set\n",
  "df.shape"
 ]
},
{
 "cell_type": "code",
 "execution_count": 80,
 "id": "709a0597",
 "metadata": {},
 "outputs": [],
 "source": [
  "#assigning values to the variables\n",
  "age_salary = df.iloc[:,1:-1].values\n",
  "iphone_purchase = df.iloc[:, 3].values"
 ]
},
{
 "cell_type": "code",
 "execution_count": 81,
 "id": "b18f2552",
 "metadata": {},
 "outputs": [],
 "source": [
  "#training the model\n",
  "from sklearn.model_selection import train_test_split as test\n",
  "as_train, as_test, ip_train, ip_test = test(age_salary, iphone_purchase, test_size = 0.25,
random_state=1)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 82,
 "id": "dbc1cfed",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
```

```
   "[0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 0 0 0\n",
   " 0 1 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0\n",
   " 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1 0 0 0]\n"
  ]
 }
],
"source": [
 "from sklearn.svm import SVC\n",
 "from sklearn.pipeline import Pipeline\n",
 "svm = Pipeline([\n",
 "(\"scale\", StandardScaler()),\n",
 "(\"svm_clf\", SVC(kernel=\"poly\", degree=3, coef0=1, C=10))\n",
 "])\n",
 "svm.fit(as_train, ip_train)\n",
 "ip_predict = svm.predict(as_test)\n",
 "print(ip_predict)"
]
},
{
"cell_type": "code",
"execution_count": 83,
"id": "eabdc54e",
"metadata": {},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "Accuracy score:  88%\n"
  ]
 }
],
"source": [
 "#printing the accuracy score of the matrix\n",
 "from sklearn import metrics\n",
 "accuracy = metrics.accuracy_score(ip_test, ip_predict)\n",
 "accuracy = \"{:.0%}\".format(accuracy)\n",
 "print('Accuracy score: ', accuracy)"
]
},
{
"cell_type": "code",
"execution_count": 84,
"id": "c94dd1d2",
"metadata": {},
"outputs": [],
"source": [
 "#ROC Curve\n",
 "from sklearn import metrics\n",
```

```
  "from sklearn.metrics import roc_curve\n",
  "fpr, tpr, thresholds = roc_curve(ip_test, ip_predict)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 86,
 "id": "fd6049ca",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAAXQAAAD4CAYAAAD8Zh1EAAAAOXRFWHRTb2Z0d2
FyZQBNYXRwbG90bGliIHZlcnNpb24zLjUuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/YYf
K9AAAACXBIWXMAAAsTAAALEwEAmpwYAAAncElEQVR4nO3deXxxU9bnH8c+THUIg7CI
QNnFXBMPmviG4Um9btVZtrb1eb8W616VW69bWWlvrfqqm11m7cXrVlEcFdVIgCiqyiYQ+LLA
ESErLMzO/+cRIcQiATmJkzy/f9evgCGyRyS50j88uM5v/Mcc84hIiLJL8PvAkREJDoU6CIiKUKBLi
KSIhToIiIpQoEuIIisvz6wl26dHF9+/b168uLiCSluXPnbnbOdW3uPb8CvW/fvsybN8+vLxcRSU
bNXXe3lPLRUQkRSjQRUREhAJdRCRFKNBFRFKEA11EJEUo0EUEAl1EJEUo0EUEAl1EJEUo0E
UEAl1EJEUo0EVEUk15dPU2JgghhxU9ryOySSTROPrNb8PvlpsXrfqm11m7cXrVlEcFdVIgCiqyiY
```

KK/VugicbFxCUwcB2vnwMDRXph36Ol3VTGnQI+xT1ZvpS4Q4oge7emUn+N3OSKpLVAHH/w
OZjwCee3hm3+Eo7+ZsMO0ok2BHmNfb1fU6lwkptbO9YZpbVwER38LznkY8tOrzalAjzEFukiM1
VXDu7+AWU9Bu4PgOxPgsHP8rsoXCvQYqqoNMG/NNjIzjGH9OvldjkjqWfG+t4OlfDkc/30YdT/
kdfC7Kt8o0GNo9spyAiHHoN6FFORl+12OSOqo2Q5v3Atz/wQd+8H3JkO/U/yuyncK9BhSu0UkBp
ZOgyk3wY4NcML1cNpdkKOBd6BAj6mZCnSR6KnaDK/dDgtfgm5HwiV/hV7H+11VQlGgx8j26n
oWrttOdqZR3Ef9c5H95hwsfBle+4k3t/y0u+CkmyBL24CbUqDHSMmKLTgHg4s60iYn0+9yRJLT9
rXw6s3wxTToeTxc+CR0P9LvqhKWAj1G1D8XOQChEHzyZ3jjHgjWw+hfwPBrIUOLo31RoMeI5
p+L7Kcty2DyDbDyfW/nygW/h079/a4qKSjQY2BTZS1Lv6okLzuD43oX+l2OSHIIBqDkaXjnIcjMg
QsehyFXps1t+9EQ0dgxMxtjZkvNrNTM7mjm/Q5mNtnMPjOzRWZ2VfRLTR4ly73V+dC+ncjJSt3J
biJR89Ui+OMoeONnMOAMuO4jOP57CvNWanGFbmaZwFPAKKAMmG1mk5xzi8MOuw5Y7Jy
7wMy6AkvN7G/OubqYVJ3gGrcrav65SAsCtfD+o96PvEL41vNw1H8oyPdTJC2XYUCpc245gJlNA
MYC4YHugAIzM6AdUA4Eolxr0pi164HQ6p+L7FXZHG/E7aYlcOwlMPqXkK9F0IGIJNB7AmvC
XpcBw5sc8yQwCVgHFACXOOdCTT+RmV0DXANQVFS0P/UmvLXbdrJySzUFuVkcfXB7v8sR
STx1VfD2Q16/vP3BcNk/4dDRfleVEiIJ9Ob+7eOavB4NzAPOAAYAb5jZ+865it1+k3PjgfEAxcXF
TT9HSmjc3TK8fyeyMtU/F9nN8ve8YVpbV0Lx1XDWz7255RIVkQR6GdA77HUvvJV4uKuAXzn
nHFBqZiuAw4GPo1JlEpm1q3+udovILju3eRc8P3P3kROg2A70+Fvif6XVXKiSTQZwMMDzawfsBa4F
LisyTGrgTOB982sO3AYsDyahSYD51xY/1y9QBEAPn8VptwMVRvhxBvgtDshu43fVaWkFFgFPdO
Rcws3HAdCATeN45t8jMrm14/1ngAeAFM1uA16K53Tm3OYZ1J6RVW6pZt72GTvk5HNa9wO9y
RPy1Y5M3f2XRK9D9aPjOP6DnEL+rSmkR3VjknJsKTG3ysWfDfr0OODu6pSWfxu2KI/p3IiND26
4kTTkH8/8J0273LoCefjecdCNk6b6pkAsaY7RaNoZkO7Rf1zSVvby7xZ5V++Dr2GesO0uh3ud1VpQ4
EeJV7/XAO5JE2FQjD3ee8pQi4EYx+6GYf+pYVpxpkCPki++2sGWqjq6t8+lf5d8v8sRiZ/NpTelg
9E//qf5g3T6tjX76rSkgI9SmaG3R1qum1Z0kEwALOehHd/CVm5MPYpoO67um3fRwr0KJml+S2S
TjYsgInXwfrP4PDz4bxHoeAgv6tKewr0KAiG3K4+4Ji+qfS0oL1MKMR+CD30GbjvDtP8ORY7UqTx
AK9ChYvK6CipoARZ3a0qunj4uvfPNSGHQZjH4I9fqXFJQ7Q54+wH46H7gQy7UqTx
wH46H+gQy+4/GU45Cy/q5JmKNCjoPGGohoPGGohMOUaBLiln2tvc4uG2rYdg1cOY9kKu7oBOVAv0A1
QVCzF5ZDmiFLilk51aYfjfM+yt0HghXTYM+I/2uSlqgQD9A88u2UV0X5JBu7ejWP7e5WPs/vckQO3JLJ
8OotULUZTroZTr0dsvW9nQwU6Adopu4OlVRR+RW8dhssnggHHeM9eOLg4/yuSlpBgX6AZsra
D3n4w 4LN/wLQ7+4/4ccappWEFOgHoKY+yNvzNrGZzG2z1yc/4ccapppWEFOgHYIOGoKY+yNvzNrGZzGD4f0U6JKEtq2GyTfCsregW79zGyXfCsregW79zGyXfCsregW79zGyXfCsregW79zGyXf
fVcl+UqAfgE9WbaAuGGLHu3pT9fPPJJDfwgZZ+yNvzNrGZzGD4f0U6JKEtq2GyTfCsregW79zGyXfCsregW79zGyXf
4SJo6DNSUw4Ey44DEoTM2HtqcbBfoB2DWQS/vPJJRkE62Hm4/Duw94L7xdAz6jm7bTyEK9P
20ozbAZ2XbycwwhvbV7c+S4NZ/5g3T2rDAm71yziNQ0N3vqiTKFOj7afakoIhx+CiQgrytBtAEl
R9Dbz3K/jwccjvAhf/BY68O+qp6/nnardIglo1CyaNgy2lcNzlMPpBb0KiPbb0KipCwF+n6atWtcr
p4fKgmmmthLevA9m//8G72HnFv2DAGGX5XJXGgQN8P26rqWLGugN8P26rqWLGugN8P26rqWLGugN+n6atWtcr
YPi1cMbPILed31VJnCjQ98PmyjqWr6vQz26rqWLGug8M9eMLg4/yuSlpBgX6fat0obSmc4T8aKcKZ
wp4fKuIr57zZK1Nv9SYknnwnrNKbhmml8kQX6ftD8s8kKhmml52a5fT8c0kIlRu8q+KXd4Dmc0kIlRu8qyIfT4Eeg+DyV6DHH7E7R5O/DZqQZ
Sxsoavty4gzbZmQzqVeh3OZKOnIN5f/NaLIFaOOs+GDkOMvW/c7rTd0ArNU5XHNqvEzlZmnshc
bZ1pfcEoeXvQtEJ3jCtLof4XZUkkCAV6K2lcrvgiFISP/wBv3QeWAec9Csf/QMO0ZDcK9Fb6ev+5
Al3iZOPnMOl6KPsYDhkF5/8OCnv7XZUkIAV6K5RtqyYxI46uLWbWbVlmoK8rI46uLWbWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8rI46uLWblmoK8rRtWbVlmoK8r
Tnt4KLxcOzFGqYlexXRv9fMbIyZLTWzUjO7Yy/Yy/HnGZm88xskZm9F90yE0Nju2V4v85kZuh/F90yE0Njuy2V4v85kZuh/Ko
mhdZ/C+NPgnfh8PPhuo9h0CUKc9mnFlfoZpYJPAWMA4YAE8xsSJPDvgncA4YA5wN/j2aRiaKx3TJ8Xmcys3g/C/HHGZm88xskZm9F90yE0Nju2V4v85kZuh/Ko
dYtRvb5S/1xirn4nvPtLmPkeS5HeDS/8H8/6nXWZWbEAIwFlgcd6XYvgOudU5d2l/C8u94OdU
AzrmN0S7Ub8457T+X2Fr5odcrL18GQ66566EUQ9Am0K/q5IWKtCjwDmXZWazgCeBTGCqc25RN
pld2dwnMrNzGyOmc3ZtGnT/lXssxWwnsWbq9hUWce6h/Y9F90yE0Njuy2V4v85kZuh/KGmc3ZtGnT/lXssxWwnsWbq9hUWce6h/Y9F90yE0Nju2V4v85kZuh/K
7YgKc2mlSFbozTXtXDOf53jgTKANMMvMSpxzX+z2m5kqH8
u0fLF6zDlRqhYByougzN+Cjrj6nzEgM5kqH8
oqtoC0+6ABf+ErofD1W9A76F+VyVJpKWy2xgoJn1M7Mc4vYUpKpKWy2xgoJn1M7Mc4vMSpxzX+z2m5kqH8
qf0IhR8kyzT+XKHAOFr4MTw2DRa/AqQq8m4Hd84FzGwcMB3IBKY65xY1OZ7vgK+d87vY9sb3/WO
bfEzKYB84EQ8JxzbmEsC4/5uYnh7CTofTfsujXua3a3tFq9dg8ujjKf0CwYNh7CTofTfsujXua3a3tFq9dg8ujjKfS0
iejGIufcVGBqk4892+T1I8Aj0Sstccws9VbnI/t3xrQPWFrLOfjkRXj9ZxCshbMfgH7XFaEnX6jop

A4wXRkeqfS2uVL/eGaa2YAX1Oggsfh84D/K5KUpQCvQWBYIiPlivQpZVCQSh5Bt5+EDKy4Pz
HYMj3NExLYkqB3oJF6yqorA3Qp3NbenVU/1wi8NVimDQO1s6FQ8fAeb+FDk1v3RCJPgV6C2b
qdn+JVKAOPvgtzPgN5LWHb/4Rjv6m5q9I3CjQWzBrV7tF2xVlH9bOhYnjYONiOObbMOZXkK/
vGYkvBfo+1AVCzF5RDng7XET2UFcN7zwEJU9Du4PgOxPgsHP8rkrSlAJ9Hz4r28bO+iCHdm9H
14Jcv8uRRLNiBkz6MWxdAcdfBaPugzzNyRf/KND3IXz/ucguNdvhjXtg7gvQsR98bzL0O8XvqkQ
U6Psyc9lmQP1zCbP0NZhyE+z4Ck64Hk67C3K0+0kSgwJ9L3bWBfl09TbMYET/Tn6XI36r2gyv3
Q4LX4JuR8Glf4Oex/tdlchuFOh7MXfVVuqCIY7u2Z7Ctjl+lyN+cQ4WvASv/QRqK70V+Uk3QZa+
JyTxKND3orHdoumKaWz7Wm+Y1hfToGcxjH0Suh3hd1Uie6VA34tZut0/fYVC8MkL8Po93hOER
v8Chl8LGZl+VyayTwr0ZlTW1DO/bDtZGcbQvuqfp5Uty7ytiKs+8HauXPA4dOrnd1UiEVGgN2P2
ynKCIceQokLa5eo/UVoIBrybg955CDJzvWd6Dr5Ct+1LUlFaNaNx/7n652liw0JvmNa6T+Gw8+C8
R6F9D7+rEmk1BXozNJArTQRq4f1HvR95hfCtP8FRF2lVLklLgd7E1qo6Fq+vICcrgyF9OvpdjsTK
mtneqnzT53DsJd4wrba6XiLJTYHeREnD7pbjizqSl61dDSmnrsp76ETJM9D+YLjs/+DQs/2uSiQqF
OhNqN2Swpa/6+1g2bYKhv4QzrzXm1sukiIU6E007j8/4RAFesrYuQ1evxs+/Qt0GgDfnwp9T/S7Kp
GoU6CH2VhRQ+nGHbTNyeTYXoV+lyPR8PmrMOVmqqNoEJ94Ip90B2W38rkokJhToYRpX50P7
diI7Uw/zTWo7NnrzVxb9C7ofA5dNgIMH+12VSEwp0MN8vf9c7Zak5RzM/1+Ydod3AfSMu72Ve
Wa235WJxJwCPczM5RrIldS2rfFmlZe+Ab2GecO0uh7md1UicaNAb7CmvJo15Ttpn5fFkQdr50NSC
YVgzh/hzZ+DC8GYh2HYf2qYlqQdBXqDWQ3bFUf070xmhu4UTBqbS2HS9bB6JvQ/HS74PXTs
43dVIr5QoDf4ev65+udJIRiAWU/AO7+E7DwY+zQcd5lu25e0pkAHnHNh+8/VP0946+d7t+2v/ww
OP98bplVwkN9VifhOgQ4s31zFVxW1dGmXw8Bu7fwuR/amvgZm/Bo+eAzadoaLX4Qjx/pdlUjCU
KDz9e3+3+I/p3xvRP9sS0+iNvVb75Cxh0GYx+SMO0RJqI6O4ZMxtjZkvNrNTM7tjHcUPNLGhm34p
eibE3S88PTVy1O2DqT+D50VC/Ey5/GS56RmEu0owWV+hmlgk8BYwCyoDZZjbJObe4meMeBq
bHotBYCYXcrh0uuiCaYErfgsk3wvY13jbEM++B3AK/qxJJWJG0XIYBpc655QBmNgEYCyxuctz1
wMvA0KhWGGOfb6hka3U9B3fIo0/ntn6XIwA7t8L0n8K8v0HngXDVa9BnpN9ViSS8SAK9J7Am
7HUZMDz8ADPrCVwEnME+At3MrgGuASgqKmprTHRuF1x5IAu6p8ngsWTYQqtULUZTroZTr
3d25YoIi2KJNCbSznX5PVjwO3OueC+QtE5Nx4YD1BcXNz0c/hC7ZYEUfmVF+RLJsFBx8B3/w9
6DPK7KpGkEkmglwG9w173AtY1OaYYYmNAQ0l2Ac80s4Jz7dzSKjJVAMMMTHK8oBGKlA94dz
MO/vMP0u76LnmffCCddrmJbIfogk0GcDA82sH7AWuBS4LPwA51y/xd+md1YmEu0owppm34p
3Qr0s+BxdqRnbcbV0FU26ZEZW9D0Ui48AnoMtjvqkSSSSouB7pwLmMtDvqkSSVouB7pwLmMt
DvqkSSVauB7pwLmMtDvqkSSVauB7pwLmMtDvqkSSVauB7pwLmMtDvqkSSVauB7pwLmMtD
PxrjGmGnsn4/or9V5XIVCMPsP8OZ93q365/4Giq+GDM2gFzkQEd1Y5JybCkxt7rpyKzeyOZt7/rpnNb/
gx08wGRb2/ewaYt0pgwWNQmBgByEWSXdreKVobCDJ7pdc/1wo9DoL18OHv4b2HIbstfO
NZGHSphmmJRFFaBvq81duoqQ9xWPcCuhbk+l1Oals3z7ttf8MCb/bKub/bKub+Bdt38rkok5aRtoM9
Hulhiq3+mtyD98HPK7wCV/hSMu8LsqkZSTtoG+Hulhiq3+mtyD98HPK7wCV/hSMu8LsqkZ
0tAz0nXVBPl29bd7ZfYfvIudV/wbBpzud1UiKS+SQG9X77btpbd7ZfYfvIudV/wbBpzud1UiKQ45
77hDdOqWAvD/xvOuBtyNV9eJF7SMtBnartidFWXw7Q7Yf4E6HIYXP069md1UiaSetA10XR
A+Qc7D43zD1Nm9C4im3eT+ytGtxA9pF+gVNfUsKNtGVoYxtK8ekrDfKjfAq7fA51Ogx3Fwxb+
8oVoi4pu0C/SPl5cTcjCkqJD83LQ7/QPnHHz6V29eebAWRt0PI66EibfA
MvfhT4nwgWPQ5dD/K5KRBqkXaA37j8fqeeHRi4UhI/Hw1v3g2XCeb+F46/SMC2RBJNWgv5eV
ceS9RXkZmUwuKjQ73KSw8bPvRuEymbDIaO8Y8YVodevldlYg0I60GvNbBPvRuEymbDIaO
8YYVodevldlYg0I60CvaRhdX58n47kZWf6XE2CC9bbPvRuEymbDIaO8YYVodevldlYg0I60Cva
RhdX58n47kZWf6XE2CC
9TBh4/BjEcgpx38xx/gmG9rmJZZIAkurQG+cf67+eQvWfuKNuP1qIRz9TRjzMLTr6ndVItKCNAt09
c/3qX4nvPMLmPUktOsOl/4DDj/X76pEJEpE+gbttewfFMV+TmZHurg9ir8vLlMOR73ir8vLlMOR73
nbENoV+VyUirZA2gtT5rudduGdavE9mZ2p2x2S00FvHkvzHkeOvaFKydB/1P9rkpE9k6kpDeE9m
52q37PLFdJhyE1Suh5Hj4PS7ICff76pEZD+lTaB/lTaB/vf9cF0Sp2gLT7oAF/4SuR8DFL0KvrrEplB
aBvqa8mrKtO+nQJpsje7T3uxz/OAcLX4bXFuK1Wk69A06+B06+B46+B46+B46+B46+B46+B4
33UFeu8YVpLp8LBQ2Dsk9D9KL+rEpN0Dsk9D9KL+rEpN0Dsk9D9KL+rEpN0Dsk9D9KL+rEpN0Dsk9D9
H+jOufQdyFW+HCb9GFa+D50H+jOvNbauXDoGxT0L3p8yFKaskzsPaDkJJ+n7m7S3XMC2RljJb
PaDkJkN5z/m7S3XMC2RlJbygR6+Ord0mEPy1WJvmNbauXDoGG8yYoeeflclInGQ+oFemibtlkAd
fPBbmPEbyGsP3/yjN4clHf4SExEgxQM9FHKUrEiDC6Jlc71V+cbF3kTEMQ9Dfor/BSYie0jpQF+y
oYJt1fX0LGxD705t/C4n+uqq4Z2HoORpaHcQfOd/4bAxflclIj5J6UCftru0NTrn++YoY3TGvrSu0NTrn
QaPugzwNHRNJZxFtezCzMWa21MxKzeyOZt7/rpnNb/gx08wGRb+U1kvJ7Yo1272tiH++YoY3TGvrSu/p
QaPugzwNHRNJZxFtezCzMWa21MxKzeyOZt7/rpnNb/gx08wGRb/U1kvJ7Yo1272tiH++ADD43h

TvKUIKc5G01+IK3cwygaeAUUAZMNvMJjnnFocdtgI41Tm31czOAcYDw2NRcKTqgyE+SrX5L
Utf84Zp7fgKTvgxnHYn5LT1uyoRSRCRtFyGAaXOueUAZjYBGAvsCnTn3Myw40sA3x86uWDtd
qrqgvTvkk+PDkneP6/a7M1fWfgydDsKLv079Bzid1UikmAiCfSewJqw12Xse/V9NfBac2+Y2TXAN
QBFRUURlrh/wvvnScs5WPB/8NrtUFsJp/8UTrxRw7REpFmRBHpzVxNdsweanY4X6Cc1975zbjxe
O4bi4uJmP0e0fP380CTdrri9DKbcDF9Oh57F3jCtbkf4XZWIJLBIAr0M6B32uhewrulBZnYs8Bxwj
nNuS3TK2z+1gSBzVm4FvAmLSSUUgrl/gjfuBReE0b+E4f+lYVoi0qJIAn02MNDM+gFrgUuBy8I
PMLMi4BXgCufcF1GvspU+Xb2N2kcIww8qoHO7XL/LidyWZd4OllUfQL9TvWFanfr5XZWIJIk
WA905FzCzccB0IBN43jm3yMyubXj/WeAeoDPwdMN+74BzzrdH4MxMtv55MAAlT8E7v4DMX
LjwSRh8uW7bF5FWiejGIufcVGBqk489G/brHwI/jG5p+29WMvXPNyyAieNg/Tw47Dw471Fo38P
vqkQkCaXcnaLVdQE+Xb2NDINh/RK4fx6ohRmPwAe/gzYd4dsvwJHf0KpcRPZbygX67JVbCYQc
g3p1oEObbL/Lad6aj71V+ealcOylMOaX0DaB//IRkaSQcoHeuF1xZCK2W+qq4K0H4KNnoX1P+O
5LMHCU31WJSIpIuUCflajzW5a9A5N/DNtWw9Afwpn3enPLRUSiJKUCffvOehau3U52plHct6Pf5
Xh2boPXfwqf/hU6DYCrXoM+J/hdlYikoJQK9I9XXlBNYycHzvrmM+J/hdlYikoJQK9I9XXlBNYycH
O8nnyohIwkqA1IuehOmf79gIU2+Dxf+G7sfAZXPg13XGYRuSdAL2N8Xxf8DcFXM3wCr9y+tt33rlz8N
kk7mJ2EmHYH1FfDGT+0B3zQKTv47HPsjGHCG3fWRSEpIqUD/ePkF1SR2fODsbf4n/ndl0BNYydz
ttf8N8OOICOPdRKOge268pIhKBlAj0xt0tw/p1IiszIzZfpL4GZvwaPngM24naGi1+EI8fG5muJiOyHl
Aj0mbFut6wu8VblW76EQZfB6Ic0TEtEEk7SB7pzLnYPhK7dAW/dDx+Phw694fKX4fKX4ZCzovs1RE
SiJOkDfXV5NWu37aSwbTZHHBTFyYVelb8Lkm2D7Ghh2Dd35D+S2i97nFxGJnFxqY4G2sqQP9F3bFt3JiMj
CrtMqsth+k/hs79D54dwg2lQNOLAP+ISIylTKBHpX++eCK8eitUb4GGpX++eCK8eitUb4GTb4FTfuJtSxQRSQJJHei7
988PYCBX5QaYeissmQwHHev1ynscG6UqRUTiI6kDvXXTjDjbvqKVbQS4Duua3/hM4B/P+DtPv9
LYlnvVzGDlOw7REJCkldaCH3+5vrb1Lc+sqmHwDLH8HikbChU9ExfuJtSxSxQRSQJJHei7
HgjD7OXjzPu9W/XN/A8VXQ0aaMbkgBMi7Z9vWuoN01rzkbef/PzfQW
FRDKsUEYmfpA30Jesr2L6znl4d29C7Y9t9Hxyshw8fg/d/DTu9Hxyshw8fg/d+DTn5cNH/wLGXaJiEg8JG2gh9/ub629S3PrKph8Ayx/B4pGwoVPRMX7ibUsUkUkCSRtoO9ILqVLhaq6R7ve+3/wfFV0NGmjG5IATIu2fb1rqDdNa85G3n/z830Fg
Z5t+1/tQCO/Aac+iTn5cNH/wLGXaJiEg8JG2gh9/ub629S3PrKph8Ayx/B4pGwoVPRMX7ibUsUkUkCSRtoO9ILqVLhaq6R7ve+3/wfFV0NGmjG5IATIu2fb1rqDdNa85G3n/z830Fg
7PyWVTO9XvmWUhh8BZz9ALTYYdtJOfgYdJCov3e7bTYYdtJOfgYdJCov3ebtYCov
gin/DgNN9q1NEJJ6SMtBbnNfTPd1udf/kGTL4RKtbBuMvhzt+0B3zQKTv47HParMnw
BdDoOrX4few3yuUEQk/iK6m8bMxpjZUjMrNbNbM7MrNbNbM7mnnfzOzozHvfzOzozHvf
78ORQWPiSN0jr2vcV5iKStlpcoZtZJvAOcA+wH3jazJ919SbJvOK6v4GVvAX2ab2yeiuFgS
08VvAX2k0ugR7HwZUT4aCjY/HlRESSRiQtl2FAqbuXuXsN8CQwNnnOK6v4GVvAX2ab2yeiuFgS
s659dEueMOcSbyZ+1PaBAIw6n4YcR1kJmXnSEQkFjLbJn0
6ZNra0VgK+ye/EZhzL7nClw4g0KcxGRBpG0gb66vJq1m7bXdG3dY2dGf8jJmfzMmNmn0
JNRRLoZUDvsNe9gHX7HtYvf/v9OAbn3Hhg/PUbn3Hhg/PEe70fiv//jbOrHE6rmeIi
9F/1xERPauxRW6cy5gGlWmyzzbZGZISFORJONsYKCZ9TOzHOzHO2czYFVzbsdhkBbI
V2+MeeDfu1A66LbmkiItIaakLiKQ1BbqQoIppQ1AgFdveBqzXxbpyeVPX9nshbqQ1PXgoT13F2FuMtJ
Bjrn9KBzTg8Hcs59nHNd3dm3vDt0A//EGG2xzlX7Hcd8R863/5/QQq3NWy0VEJUo0EVEUkSyB
vp4wvwbrK2r9K1h+hIEwp0EZEUkRSBrqZN2pyGaiy0IBy1K+r+giIrKrKTYB
D/qjKaWjnsuKFmFjSzb8WzvliI5I5JZN7NDOzm2dmi8zsWjJXJvZvZXJzvZZwzkn9dRRMw3
ezDaa2cK9vB/9/HLOJeQPvFG9y4D+QA7wGXBk2POBV7De2LSCOAjv+uOYjt/0qjY/Fk0kZ7
XDOYce9jTf181t+1x2HP+dCvOf2FjW87uZ33XE457uLar-6Z+nmFvUvh1M65C25mc25rw8sSvKdDJbNI/pwBrgdeBjb
Gs7gYieScLwNecc6tBnDOJft5R3LODigwMwKuAnF6IL0ag+nTiKtP
Z+r8f6GT2YYvOZ9vrnOZ9vrvZ9vvZoqXOR/3wo0NHM3v4DyN9wjrmzuw4XiU54
vop5fET3gwidRezh1Eon4fMzsdLxAPymmFcVeOff9Ld6SXiTnnAUcD5wJiU54
c+6LWBcXI5Gc82hgHnAGAMAB4w8zed85VxLg2v0Q/6LWB8I5Gc82hgHnAGA48+zed85VxLg2vTed85VxLg2v
XGqLVYiOediYEJDmHcBzjJo/d7e7JyrAqrMbMmmMphDDZN/bE4
ma0ADgc+jk+JcRf1/Erklks6Ppy6xXM2syLgFJF6thWxnJ1z/ZxzfYGXgB8lPQZ8/hE4
GTzSzLzNoCw4Elca4pifbp5JUBn4vxJ2he7T8OHUEZ7zPUBn4mGFWv

AJfGkugjPOaVEcs7OuSVmNg2YD4SA55xzzW5/SwYR/jk/ALxgZgvw2hG3O+eSdqyumf0DOA3o
YmZlwL1ANsQuv3Trv4hIikjklouIiLSCAl1EJEUo0EVEUoQCXUQkRSjQRURShAJdRCRFKNBF
RFLE/wMsjMOVF6IMDQAAAABJRU5ErkJggg==\n",
         "text/plain": [
          "<Figure size 432x288 with 1 Axes>"
         ]
        },
        "metadata": {
         "needs_background": "light"
        },
        "output_type": "display_data"
       }
      ],
      "source": [
       "#Plot\n",
       "import matplotlib.pyplot as plot\n",
       "plot.plot(fpr, tpr, linewidth=2)\n",
       "plot.plot([0.0, 1.0], [0.0, 1.0]) \n",
       "plot.show()"
      ]
     },
     {
      "cell_type": "code",
      "execution_count": null,
      "id": "a92ec84b",
      "metadata": {},
      "outputs": [],
      "source": []
     }
    ],
    "metadata": {
     "kernelspec": {
      "display_name": "Python 3 (ipykernel)",
      "language": "python",
      "name": "python3"
     },
     "language_info": {
      "codemirror_mode": {
       "name": "ipython",
       "version": 3
      },
      "file_extension": ".py",
      "mimetype": "text/x-python",
      "name": "python",
      "nbconvert_exporter": "python",
      "pygments_lexer": "ipython3",
      "version": "3.9.12"
     }
    },

"nbformat": 4,
"nbformat_minor": 5
}
**Report:**

Step 1: read the csv into the dataframe using the pd.read_csv() function

Step 2: print the shape of the dataframe

Step 3: Assign the variable X to the values of Age and Salary and variable y to the values of whether the iphone was purchased or not

Step 4: Train the model based on the 75-25 split

Step 5: pipeline

Step 6: Calculate the accuracy score

Step 7: Calculate the ROC Curve

Step 8: Plot the curve