Assignment #2 **Possible Points: 100**
CS-770 ML **Due date: 31st October 2022**

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 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
K9AAAACXBIWXMAAAsTAAALEwEAmpwYAAAncElEQVR4nO3deXxU9bnH8c+THUIg7CI
QNnFXBMPmviG4Um9btVZtrb1eb8bWWlvrfqm11m7cXrVlEcFdVIgCiqyiYQ+LLA
ESErLMzO/+cRIcQiATmJkzy/f9evGCyRyS50j88uM5Mv5v/Mcc84hIiLJL8PvAkREJDoU6CIiKUKBLi
KSIhToIiIpQoEuIpIisvz6wl26dHF9+uPd8uLiCSluXPnbnbbOdW3uPd8CvW/fvsyZM8evLy8MB
bNXe3lPLRUQkRSjQRURShAJdRCFKNBFRFKEAl1EJEW0GOhm9rytZbTS2ryzbTSzMrNb
P5ZjYk+mWKiEhLIlmhvwCM2cf75wCCLzpvDW2J
mhWbWwzm3PlpFiogkGucctYEQO2oD7Kg4sKM2QGXDzztq69lEQGqa2o4dfP/Un7
QSK777sVRrykaNxb1BNaEvS5r+NgegW5m1+Ct4ikqKorClxYRaR3nHDvrg3xjcJ492
P9Y6pD0b+PImjbCUPZ4/n6IyVvFRfASRmoFszH2v2LJ1z44HxAMXFxXqyohELBRyVNU1H7C
7v65vEtBfB3ZjEIeilD7ZmUZBXjbtcrO8H3lZFDT83Pi6MDvIiWv/xNEr/0R9Tkc+H/4URx95YXQ
KaCIagV4G9A573QtYF4XPKyIpIpIBAMUVUbpLIhaPc3jHfUBqJWU152Bu1ysyloDN69hHFBXrb3
sbCPF4S9n5uVue8vtLoEJo6DLV/CcZeTO/pBDm/TMWrn0VQ0An0n0TMM7qqw7DDGe3qn4skv7qw/
nBlQ0/Y6w/vLXB3D+zKGu/XO+uDUaspPyczLHD3HrYFedl7BnTD+/m5WWRnxnjdm0lvHU/f
PwH6NAbLn8FDjkztl+TCALdzP4BnAZ0MbMy4F4gvB7ZFvjhbrK3FvjhbrK3
Va/rQvjHbUBKmoC1AVCUanJDC9Md1v9Nh/GXiBnN7sazs/JIjOjuQ5vgil9EybfCNvLPh/wRk/g
9x2cfnSkexy+U4L7zvguqhVJJKGnHNU1wWbCdz6PQO4uXZFWGi35kLdvmRm2G4tiYKwMN7t
dQth3DYnE7MkCOIDVV0O038Kn/0duhwP0duhwKP5gGRSPiWoJv43NFUkGw8ULdHjslIgtjbzVcT1UU
L9TlZGU022r4eejWcHdaaCO8f7x7GuVkZ6RHE0bB4xrx6K1RvgZMNvhVNug+y8y8uJehQJe0FAi
wbbubiXNOLd00Cu7Kmnnqq66PWH22RnNtv33f1j23aimgaxcB6ApOGG3aimgaxgV5e+6mKMjNJj83k6xY
6iT6HEO5v0Npt8F9TVw1s9WYQ6W+kKtAl5pxz1NSHdr84t9fVb/2egR22Y6Y6IuGJ0gzmi8UNe4
hzivSQDvcbEuu5mA9i7UZSTDhTqJnq2rYPWkdLvmEnnmFZWA
770Vsfs+4safg1FqEGc1XqhrXPG2GMbZTfrE3s9tstPkQp1ETyjobUN8635v6865v4HiqyEjcf5lpUB
PQcGQ263N0OzFuWbbFWH7iGsC7KgL4KJ0oS43K6PJhbndL841u494j4t5ulAnPtm0FCZdD2s+g
kPOgvMfg8LeLf62eFOgJ5D6YKjZIT+RhnHjx6uj8ujeKGubU6m83OzyMlKnFWMSMSC9
fDhY/DeryEnHy4aD8de7K3QE5ACPY521AZ4ZNrnrNte06Rd4xVxbTRv5MjvWx8W72De87jJve
1NE2O1MX6kT2V/1OePdXMPMJyO8Kl/wNjjjf76oiokCPo5nLLZNPmNqwWaaWPwnaJ0oS43K6PJhbndL841u494j4t5ulAnPtm0FCZdD2s+g
kPOgvMfg8LeLf62eFOgJ5D6YKjZIT+RhnHjx6ujeKGubU5m83fRtSKM83OzyMlKnFWMSMSC9
fDhY/DeryEnHy4aD8de7K3QE5ACPY521AZ4ZNrnrNte06Rd4xVxbTRv5MjZvRWx+8W6Zu6i27
VzIjusP6wLdZLG1n0KE6+HrxbAURBOY9Au65+V7VPCvQ4enluGX+72De87jJve
1NE2O1MX6kT2V/1OePdXMPMJyO8Kl/wNjjjf76oiokCPo5nLNgPwnyf347TwY/TDuu1xU0detvrDIr5a
+aHXKy9fBoOvgLMfhDaFflcVMQV6nARDjskL11HZVEDbrRBhwDJVLF/PLVtpVv/8zUiVkBflXr
xHZYCXFF7YDXvfbpDLIhN/Cq38AwLCv6rEhcaEbvTNe7nmLCoh/d/6Lh3xBIQ9tDKTBjYD/TGiYFYXSxMCQF2BVRFXzhnuqKDx5/9Wg
D/NL+rajUFepwsWV/B9p319OrYRmEukki+fMMbplwWxFkb8CM6427sAmoQU6HHS2G45x6YUBnn
ysREcAbpjXtTpg/AboeDle/Ab2H+l3VAVGgx8msZVsAOyGAF58rEUlzzsGif8x
8C2Tl+l3ZAVOgx0F9MMTHK7z++Uit0EX8U7EeXr0fIr4KBw+GCyfCQUf7XVXUKNDjYH7Zd
qrqggzomk/v+/9vEfqSmS9pyDT/8C0++GYC2MesDrl/s8TCvaUutsEtSshv65VuciPihfAZN/DCtmQN
T4MLHofMAv6uKFcV6HMxU/1kJB+Oh/4O0HwDLh/N/BkO8n1DCtaFOgx1hNZA5q7YCM

KK/VugicbFxCUwcB2vnwMDRXph36Ol3VTGnQI+xT1ZvpS4Q4oge7emUn+N3OSKpLVAHH/w
OZjwCee3hm3+Eo7+ZsMO0ok2BHmNfb1fU6lwkptbO9YZpbVwER38LznkY8tOrzalAjzEFukiM1
VXDu7+AWU9Bu4PgOxPgsHP8rsoXCvQYqqoNMG/NNjIzjGH9OvldjkjqWfG+t4OlfDkc/30YdT/
kdfC7Kt8o0GNo9spyAiHHoN6FFORl+12OSOqo2Q5v3Atz/wQd+8H3JkO/U/yuyncK9BhSu0UkBp
ZOgyk3wY4NcML1cNpdkOBd6BAj6mZCnSR6KnaDK/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+udovILju3eRc8P3kROg2A70+Fvif6XVXKiSTQZwMDzawfsBa4F
LisyTGrgTOB982sO3AYsDyahSYD51xY/1y9QBEAPn8VptwMVRvhxBvgtDshu43fVaWkFgPdO
Rcws3HAdCATeN45t8jMrm14/1ngAeAFM1uA16K53Tm3OYZ1J6RVW6pZt72GTvk5HNa9wO9y
RPy1Y5M3f2XRK9D9aPjOP6DnEL+rSmkR3VjknJsKTG3ysWfDfr0OODu6pSWfxu2KI/p3IiND26
4kTTkH8/8J0273LoCefjecdCNk6pkAsaY7RaNoZkO7Rf1zSVvby7xZ5V++Dr2GesO0uh3ud1VpQ4
EeJV7/XAO5JE2FQjD3ee8pQi4EYx6GYf+pYVpxpkCPki++2sGWqjq6t8+lf5d8v8sRiZ/NpTDpelg
9E/qf5g3T6tjX76rSkgI9SmaG3R1qum1Z0kEwALOehHd/CVm5MPYpOO67um3fRwr0KJml+S2S
TjYsgInXwfrP4PDz4bxHoeAgv6tKewr0KAiG3K4Ji+qfS0oL1MKMR+CD30GbjvDtP8ORY7UqTx
AK9ChYvK6CipoARZ3a0qujnj4uKWr1R16vfPNSGHQZjH4I2up5uYlEgR4Fu7Yr9tfqXFJQ7Q54+
wH46H+gQy+4/GU45Cy/q5JmKNCjoPGGohMOUaBLiln2tvc4uG2rYdg1cOY9kKu7oBOVAv0A1
QVCzF5ZDmiFLilk51aYfjfM+yt0HghXTYM+I/2uSlqgQD9A88u2UV0X5JBu7ejWPs/vckQO3JLJ
8OotULUZTroZTr0dsvW9nQwU6Adopu4OlVRR+RW8dhssnggHHeM9eOLg4/yuSlpBgX6AdLu/J
D3n4LN/wLQ7oX6n1yc/4ccappWEFOgHoKY+yNzVWzGD4f0U6JKEtq2GyTfCsreg9wi48Anoeqj
fVcl+UqAfgE9WbaUuEOLIHu3pmJ/jdzkikQuFYPZz8ObPvdfnPAJDfwgZemxiMlOgHwD1zyUpbf
4SJo6DNSUw4Ey44DEoTM2HtqcbBfoB2DWQS/vPJRkE62Hm4/Duw94j4L7xDAz6jm7bTyEK9P
20ozbAZ2XbycwwhvbV7c+S4NZ/5g3T2rDAm71yziNQ0N3vqiTKFOj7afaKcoIhx+CiQgrytBtAEl
R9Dbz3K/jwccjvAhf/BY680O+qJEYU6Pvp6/nnardIglo1CyaNgy2lcNzlMPpBb0KipCwF+n6atWtcr
p4fKgmmthLevA9m/8G72HnFv2DAGX5XJXGgQN8P26rrWLSugpzMDI7voxWPJJDSN7195dvL
YPi1cMbPILed31VJnCjQ90PJ8nKcg8FFheRl6yG4kgCqy2H6Xd4dn10OhR9Mh6LhflclcaZA3w+z
wp4fKuIr57zZK1Nv9SYknnwrnHKbhmmlKQX6ftD8c0kIlRu8qYifT4Eeg+DyV6DHsX5XJT5SoLf
Sxsoavty4gzbZmQzqVeh3OZKOnIN5f/NaLIFaOOs+GDkOMvW/c7rTd0ArNU5XHNqvEzlZmnshc
bZ1pfcEoeXvQtEJ3jCtLof4XZUkCAV6K2lcrvgiFISP/wBv3QeWAec9Csf/QMO0ZDcK9Fb6ev+5
Al3iZOPnMOl6KPsYDhkF5/8OCnv7XZUkIAV6K5RtrWbVlmoK8rI46uAOfpcjqS5YDx88BjN+D
Tnt4KLxcOzFGqYlexXRv9fMbIyZLTWzUjO7Yy/HnGZm88xskZm9F90yE0Nju2V4v85kZuh/Ko
mhdZ/C+NPgnQfh8PPhuo9h0CUKc9mnFlfoZpYJPAWMAsqA2WY2yTm3OOyYQuBpYIxzbrWZ
dYtRvb5S/1xirn4nvPtLmPkE5HeDS/8Oh5/nd1WSJCJpuQwDSp1zywHMbAIwFlgcdsxlwCvOudU
AzrmN0S7Ub8457T+X2Fr5odcrL18GQ66EUQ9Am0K/q5IkEknLpSewJux1WcPHwh0KdDSzd81sr
pld2dwnMrNrzGyOmc3ZtGnT/lXskxWbq9hQUUPn/BwO7VbgdzmSSmoqYMrN8MK5EArAlRO9
7YgKc2mlSFbozTXtXDOf53jgTKANMMvMSpxzX+z2m5wbD4wHKC4ubvo5Elrj6nzEgM5kqH8
u0fLF6zDlRqhYByOugzN+Cjn5flclSSqSQC8DwvdI9QLWNXPMZudcFVBlZjOAQcAXpAhtV5S
oqtoC0+6ABf+ErofD1W9A76F+VyVJLpKWy2xgoJn1M7Mc4FJgUpNjJgInm1mWmbUFhgNLolu
qf0IhR8kyzT+XKHAOFr4MTw2DRa/AqXfAf81QmEtUtLhCd84FzGwcMB3IBJ53zi0ys3sb3n/WO
bfEzKYB84EQ8JxzbmEsC4+nLzZWsqWqjh4d8ujbua3f5UiyqlgPr94MS6fCwYNh7CTofpTfVUkK
iejGIufcVGBqk4892+T1I8Aj0Sstccws9VbnI/t3xrQPWFrLOfjkRXj9ZxCshbMfhOH/rWFaEnX6jop

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+lyPR8PmrMOVmqNoEJ94Ip90B2W38rkokJhToYRpX50P7
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+I/p3xvRP9sS0+iNvVb75Cxh0GYx+SMO0RJqI6O4ZMxtjZkvNrNTM7tjHcUPNLGhm34p
eibE3S88PTVy1O2DqT+D50VC/Ey5/GS56RmEu0owWV+hmlgk8BYwCyoDZZjbJObe4meMeBq
bHotBYCYXcrh0uuiCaYErfgsk3wvY13jbEM++B3AK/qxJJWJG0XIYBpc655QBmNgEYCyxuctz1
wMvA0KhWGGOfb6hka3U9B3fIo0/ntn6XIwA7t8L0n8K8v0HngXDVa9BnpN9ViSS8SAK9J7Am
7HUZMDz8ADPrCVwEnME+At3MrgGuASgqKmptrTHRuF1x5IAu6p8ngsWTYOqtULUZTroZTr
3d25YoIi2KJNCbSznX5PVjwO3OueC+QtE5Nx4YD1BcXNz0c/hC7ZYEUfmVF+RLJsFBx8B3/w9
6DPK7KpGkEkmglwG9w173AtY1OaYYmNAQ5l2Ac80s4Jz7dzSKjJVAMMTHK8oBGKlA94dz
MO/vMP0u76LnmffCCddrmJbIfogk0GcDA82sH7AWuBS4LPwA51y/xl+b2QvAlEQPc4CF6yqorA
3Qr0s+BxdqRnbcbV0FU26EZW9D0Ui48AnoMtDvqkSSVouB7pwLmNk4vN0rmcDzzrlFZnZtw/v
PxrjGmGnsn4/or9V5XIVCMPsP8OZ93q365/4Giq+GDM2gFzkQEd1Y5JybCkxt8rFmg9w59/0DLy
s+1D/3waYvvGFaa0pgwJlwwWNQmBgXyEWSXdreKVobCDJ7pdc/1wo9DoL18OHv4b2HIbstfO
NZGHSphmmJRFHaBvq81duoqQ9xWPcCuhbk+l1Oals3z7ttf8MCb/bKub+Bdt38rkok5aRtoDfOb9
Hulhiq3+mtyD98HPK7wCV/hSMu8LsqkZSVtoG+a1yuAj02Vs3yVuVbSmHw5XD2g9Cmo99ViaS
0tAz0nXVBPl29lQyD4eqfR1dtpbd7ZfYfvIudV/wbBpzud1UiaSEtA33OqnLqg45jenagQxvdwBI1X
77hDdOqWAvD/xvOuBtyNV9eJF7SMtBnartidFWXw7Q7Yf4E6HIYXP069B7md1UiaSetA10XR
A+Qc7D43zD1Nm9C4im3eT+ytGtIxA9pF+gVNfUsKNtGVoYxtK8ekrDfKjfAq7fA51Ogx3Fwxb+
8oVoi4pu0C/SPl5cTcjCkqJD83LQ7/QPnHHz6V29eebAWRt0PI66DTP23FPFb2v1fqP75Adi6EibfA
MvfhT4nwgWPQ5dD/K5KRBqkXaA37j8fqeeHRi4UhI/Hw1v3g2XCeb+F46/SMC2RBJNWgV5eV
ceS9RXkZmUwuKjQ73KSw8bPvRuEymbDIaO8YVodevldlYg0I60CvaRhdX58n47kZWf6XE2CC
9TBh4/BjEcgpx38xx/gmG9rmJZIAkurQG+cf67+eQvWfuKNuP1qIRz9TRjzMLTr6ndVItKCNAt09
c/3qX4nvPMLmPUktOsOl/4DDj/X76pEJEJpE+gbttewfFMV+TmZHNurg9/lJJ6VH3ir8vLlMOR73
nbENoV+VyUirZA2gT5rudduGdavE9mZ2p2xS00FvHkvzHkeOvaFKydB/1P9rkpE9kPaBPrM0sb9
52q37PLFdJhyE1Suh5Hj4PS7ICff76pEZD+lTaB/vf9cF0Sp2gLT7oAF/4SuR8DFL0KvYr+rEpEDlB
aBvqa8mrKtO+nQJpsje7T3uxz/OAcLX4bXfuK1Wk69A06+BbJy/K5MRKIgLQK9cbviiP6dyMhI0
33UFeu8YVpLp8LBQ2Dsk9D9KL+rEpEoSpNAT+P+uXPwyZ/h9Z9BsN57FNyIH0GGbqwSSTUp
H+jOufQdyFW+HCb9GFa+D31Phgt+D50H+F2ViMRIygf6sk072FRZS5d2uRzSLU0ehxYKQskz8
PaDkJkN5z/m7S3XMC2RlJbygR6+Ord0mEPy1WJvmNbauXDoGG8yYoeeflclInGQ+oFemibtlkAd
fPBbmPEbyGsP3/yjN4clHf4SExEgxQM9FHKUrEiDC6Jlc71V+cbF3kTEMQ9Dfor/BSYie0jpQF+y
oYJt1fX0LGxD705t/C4n+uqq4Z2HoORpaHcQfOd/4bAxflclIj5J6UCftezru0NTrn++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/hdlYikoJQK9I9XlBNycHzvQtrmJMCpLZkCr94CVZvgpJvg1Nsh
O8nnyohIwkqA1IuehOmf79gIU2+Dxf+G7sfAZRPg4MH+1iQiKS+lAt33/rlz8NkEmHYH1FfDGT+
DE2+AzATdbSMiKSVlAn3zjlo+31BJblYGg4sK41/AtjUw5UYofRN6DfOGaXU9LP51iEjaSplAL2
l4mMXQvp3IzYrjIKtQCOb8Ed78ubdCP+fX3i4WDdMSkThLmUD3ZX7L5i+953qungX9T/eGaXX
sE7+vLyISJmUCvSSe/fNgPcx8At79FWTnwdin4bjLdNu+iPgqJQJ9/fadLN9cRbvcLI7pGeOHJa//zL
ttf8N8OOICOPdRKOge268pIhKBlAj0xt0tw/p1IiszIzZfpL4GZvwaPngM2naGi1+EI8fG5muJiOyHl
Aj0mbFut6wu8VblW76EQZfB6Ic0TEtEEk7SB7pzLnYPhK7dAW/dDx+Phw694fKX4ZCzovs1RE
SiJOkDfXV5NWu37aSwbTZHHBTFYVelb8Lkm2D7Ghh2DZx5D+S2i97nFxGJsqQP9F3bFft3JiMj
CrtMqsth+k/hs79D54Hwg2lQNOLAP6+ISIylTKBHpX++eCK8eitUb4GTb4FTfuJtSxQRSQJJHei7
988PYCBX5QaYeissmQwHHev1ynscG6UqRUTiI6kDvXTjDjbvqKVbQS4Duua3/hM4B/P+DtPv9
LYlnvVzGDlOw7REJCkldaCH3+5vrb1Lc+sqmHwDLH8HikbChU9Al4ExqFJEJD6SPNC9+eet6p+
HgjD7OXjzPu9W/XN/A8VXQ0aMbkgSEYmTpA30YMhRsrwcgBMi7Z9vWuoN01rzkbef/PzfQW
FRDKsUEYmfpA30Jesr2L6znl4d29C7U9t9Hxyshw8fg/d+DTn5cNH/wLGXaJiWiKSUpA30iNst6+
Z5t+1/tQCO/Aac+wi06xbz+kRE4i2JA71x//le2i31O73xtjOfgPwucMlfvemIIiIpKikDvT4YYvYKr3/e
7PyWVTO9XvmWUhh8BZz9ALTpGOcqRUTiKykDfX7Zdqrqggzomk/39mF3ctZUwFv3ebtYCov
gin/DgNN9q1NEJJ6SMtBnNfTPd1udf/kGTL4RKtbCiB/BGXd7F0BFRNJEUgb6bv3z6nKYdifMnw
BdDoOrX4few3yuUEQk/iK6m8bMxpjZUjMrNbM7mnnfzOzxhvfnm9mQ6JfqqakPMmfVVsBxSt3
78ORQWPiSN0jr2vcV5iKStlpcoZtZJvAUMAooA2ab2STn3OKww84BBjb8GA480/Bz1H2yeiuFgS
08VvAX2k0ugR7HwZUT4aCjY/HlRESSRiQtl2FAqXNuOYCZTQDGAuGBPhZ40TnngBIzKzSzH
s659dEueMOcSbyZ+1PaBAIw6n4YcR1kJmXnSEQkqiJpufQE1oS9Lmv4WGuPwcyuMbM5ZjZn0
6ZNra0VgK+ye/EZhzL7nClw4g0KcxGRBpGkYXP3x7v9OAbn3HhgPEBxcfEe70fiv//jbOrH6rmeIi
JNRRLoZUDvsNe9gHX7cUzUZGdqMqKISFORJONsYKCZ9TOzHOBSYFKTYyYBVzbsdhkBbI
9F/1xERPauxRW6cy5gZuOA6UAm8LxzbpGZXdvw/rPAVOBcoBSoBq6KXckiItKciK4oOuem4o
V2+MeeDfu1A66LbmkiItIaakaLiKQIBbqISIpQoIuIpAgFuohIijDveqYPX9hsE7BqP397F2BzFMtJ
Bjrn9KBzTg8Hcs59nHNdm3vDt0A/EGY2xzlX7Hcd8aRzTg865/QQq3NWy0VEJEUo0EVEUkSyB
vp4vwvwgc45Peic00NMzjkpe+giIrKnZF2hi4hIEwp0EZEUkdCBnkgPp46XCM75uw3nOt/MZprZI
D/qjKaWzjnsuKFmFjSzb8WzvliI5JzN7DQzm2dmi8zsvXjXGG0RfG93MLPJZvZZwzkn9dRWM3v
ezDaa2cK9vB/9/HLOJeQPvFG9y4D+QA7wGXBkk2POBV7De2LSCOAjv+uOwzmfAHRs+PU56
XDOYce9jTf181t+1x2HP+dCvOf2FjW87uZ33XE457uAhxt+3RUoB3L8rv0AzvkUYAiwcC/vRz2/
EnmFvuvh1M65OqDx4dThdj2c2jlXAhSaWY94FxpFLZ6zc26mc25rw8sSvKdDJbNI/pwBrgdeBjb
Gs7gYieScLwNecc6tBnDOJft5R3LODigwMwPa4QV6IL5lRo9zbgbeOexN1PMrkQM9ag+nTiKtP
Z+r8f6GT2YtnrOZ9QQuAp4lNUTy53wo0NHM3jWzuWZ2Zdyqi41IzvlJ4Ai8x1cuAG5wzoXiU54
vop5fET3gwidRezh1Eon4fMzsdLxAPymmFcVeJOf8GHC7cy7oLd6SXiTnnAUcD5wJtAFmmVmJ
c+6LWBcXI5Gc82hgHnAGMAB4w8zed85vLg2v0Q9vxI50BPu4dRxENH5mNmxwHPAOc65L
XGqLVYiOediYEJDmHcBzjWzgHPu33GpMPoi/d7e7JyrAqrMbAYwCEjWQI/knK8CfuW8BnOp
ma0ADgc+jk+JcRf1/Erklks6Ppy6xXM2syLgFeCKJF6thWvxnJ1z/ZxzfZ1zfYGXgB8cGXz8lcZhDZN/bE4
GTzSzLzNoCw4Elca4zmiI559V4/yLBzLoDhwHL41plfEU9vxJ2he7S8OHUEZ7zPUBn4OmGFWv

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