```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 145,
   "id": "a973a6d0",
   "metadata": {},
   "outputs": [],
   "source": [
    "#1.Implement Naïve Bayes method using scikit-learn library\n",
    "#Use dataset available with name glass\n",
    "#Use train_test_split to create training and testing part\n",
    "#Evaluate the model on test part using score and
classification_report(y_true, y_pred)\n",
    "\n",
    "import pandas as pd\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.naive_bayes import GaussianNB\n",
    "from sklearn.metrics import classification_report,
accuracy_score\n",
    "import warnings\n",
    "from sklearn import metrics"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 146,
   "id": "b8620fb6",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "RangeIndex: 214 entries, 0 to 213\n",
      "Data columns (total 10 columns):\n",
      " #   Column  Non-Null Count  Dtype  \n",
      "---  ------  --------------  -----  \n",
      " 0   RI      214 non-null    float64\n",
      " 1   Na      214 non-null    float64\n",
      " 2   Mg      214 non-null    float64\n",
      " 3   Al      214 non-null    float64\n",
      " 4   Si      214 non-null    float64\n",
      " 5   K       214 non-null    float64\n",
      " 6   Ca      214 non-null    float64\n",
      " 7   Ba      214 non-null    float64\n",
      " 8   Fe      214 non-null    float64\n",
      " 9   Type    214 non-null    int64  \n",
      "dtypes: float64(9), int64(1)\n",
      "memory usage: 16.8 KB\n"
     ]
    }
   ],
   "source": [
    "#importing the given dataset glass.csv\n",
    "dst_Data = pd.read_csv(\"glass.csv\")\n",
    "\n",
```

```
    "dst_Data.info()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 133,
   "id": "588ed4c8",
   "metadata": {},
   "outputs": [],
   "source": [
    "#splitting the dataset which is excluding last columns\n",
    "\n",
    "X = dst_Data.iloc[:, :-1]\n",
    "y = dst_Data.iloc[:, -1]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 134,
   "id": "891003a9",
   "metadata": {},
   "outputs": [],
   "source": [
    "#splitting the dataset into train and test datasets\n",
    "\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 135,
   "id": "388a7255",
   "metadata": {},
   "outputs": [],
   "source": [
    "#creating a Gaussian Naive Bayes model\n",
    "\n",
    "gn = GaussianNB()\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 136,
   "id": "b9e270c2",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<style>#sk-container-id-2 {color: black;background-color:
white;}#sk-container-id-2 pre{padding: 0;}#sk-container-id-2 div.sk-
toggleable {background-color: white;}#sk-container-id-2 label.sk-
toggleable__label {cursor: pointer;display: block;width: 100%;margin-
bottom: 0;padding: 0.3em;box-sizing: border-box;text-align: center;}#sk-
container-id-2 label.sk-toggleable__label-arrow:before {content:
\"▸\";float: left;margin-right: 0.25em;color: #696969;}#sk-container-id-
2 label.sk-toggleable__label-arrow:hover:before {color: black;}#sk-
```

container-id-2 div.sk-estimator:hover label.sk-toggleable__label-arrow:before {color: black;}#sk-container-id-2 div.sk-toggleable__content {max-height: 0;max-width: 0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-2 div.sk-toggleable__content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-color: #f0f8ff;}#sk-container-id-2 input.sk-toggleable__control:checked~div.sk-toggleable__content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-2 input.sk-toggleable__control:checked~label.sk-toggleable__label-arrow:before {content: \"▾\";}#sk-container-id-2 div.sk-estimator input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 div.sk-label input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px, 1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position: absolute;width: 1px;}#sk-container-id-2 div.sk-estimator {font-family: monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-2 div.sk-estimator:hover {background-color: #d4ebff;}#sk-container-id-2 div.sk-parallel-item::after {content: \"\";width: 100%;border-bottom: 1px solid gray;flex-grow: 1;}#sk-container-id-2 div.sk-label:hover label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-2 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-2 div.sk-item {position: relative;z-index: 1;}#sk-container-id-2 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-2 div.sk-item::before, #sk-container-id-2 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-2 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-2 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-2 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-2 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-2 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-2 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-2 div.sk-label-container {text-align: center;}#sk-container-id-2 div.sk-container {/* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 */display: inline-block !important;position: relative;}#sk-container-id-2 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-2\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>GaussianNB()</pre>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. <div class=\"sk-container\" hidden><div class=\"sk-item\"><div class=\"sk-estimator sk-

```
toggleable\"><input class=\"sk-toggleable__control sk-hidden--visually\"
id=\"sk-estimator-id-2\" type=\"checkbox\" checked><label for=\"sk-
estimator-id-2\" class=\"sk-toggleable__label sk-toggleable__label-
arrow\">GaussianNB</label><div class=\"sk-
toggleable__content\"><pre>GaussianNB()</pre></div></div></div></div></di
v>"
      ],
      "text/plain": [
       "GaussianNB()"
      ]
     },
     "execution_count": 136,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "#fitting train data\n",
    "\n",
    "gn.fit(X_train, y_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 137,
   "id": "a32ee979",
   "metadata": {},
   "outputs": [],
   "source": [
    "#predicting the test dataset\n",
    "\n",
    "y_pred = gn.predict(X_test)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 138,
   "id": "69b2620e",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Accuracy:  37.2093023255814\n"
     ]
    }
   ],
   "source": [
    "#evaluate the model on the test dataset\n",
    "\n",
    "print(\"Accuracy: \", accuracy_score(y_test, y_pred)*100)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 139,
   "id": "a756480b",
```

```
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Classification Report:               precision    recall  f1-score
support\n",
      "\n",
      "           1       0.19      0.44      0.27         9\n",
      "           2       0.33      0.16      0.21        19\n",
      "           3       0.33      0.20      0.25         5\n",
      "           5       0.00      0.00      0.00         2\n",
      "           6       0.67      1.00      0.80         2\n",
      "           7       1.00      1.00      1.00         6\n",
      "\n",
      "    accuracy                           0.37        43\n",
      "   macro avg       0.42      0.47      0.42        43\n",
      "weighted avg       0.40      0.37      0.36        43\n",
      "\n"
     ]
    }
   ],
   "source": [
    "print(\"Classification Report:\", classification_report(y_test,
y_pred))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 140,
   "id": "935c3f0d",
   "metadata": {},
   "outputs": [],
   "source": [
    "#2 Implement linear SVM method using scikit-learn\n",
    "#Use the same dataset above\n",
    "#Use train_test_split to create training and testing part\n",
    "#Evaluate the model on test part using score and
classification_report(y_true, y_pred)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 141,
   "id": "35ba4238",
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.svm import SVC\n",
    "from sklearn.metrics import classification_report,
accuracy_score\n",
    "import warnings\n",
    "from sklearn import metrics"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 142,
   "id": "6cfb0842",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "RangeIndex: 214 entries, 0 to 213\n",
      "Data columns (total 10 columns):\n",
      " #   Column  Non-Null Count  Dtype  \n",
      "---  ------  --------------  -----  \n",
      " 0   RI      214 non-null    float64\n",
      " 1   Na      214 non-null    float64\n",
      " 2   Mg      214 non-null    float64\n",
      " 3   Al      214 non-null    float64\n",
      " 4   Si      214 non-null    float64\n",
      " 5   K       214 non-null    float64\n",
      " 6   Ca      214 non-null    float64\n",
      " 7   Ba      214 non-null    float64\n",
      " 8   Fe      214 non-null    float64\n",
      " 9   Type    214 non-null    int64  \n",
      "dtypes: float64(9), int64(1)\n",
      "memory usage: 16.8 KB\n"
     ]
    }
   ],
   "source": [
    "#importing the given dataset glass.csv\n",
    "dst_Data = pd.read_csv(\"glass.csv\")\n",
    "\n",
    "dst_Data.info()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 143,
   "id": "da7acb68",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Accuracy:  51.162790697674424\n",
      "Classification Report:                precision    recall  f1-
score   support\n",
      "\n",
      "           1       0.36      0.89      0.52         9\n",
      "           2       0.58      0.37      0.45        19\n",
      "           3       0.00      0.00      0.00         5\n",
      "           5       0.50      0.50      0.50         2\n",
      "           6       0.00      0.00      0.00         2\n",
      "           7       0.86      1.00      0.92         6\n",
```

```
      "\n",
      "    accuracy                           0.51        43\n",
      "   macro avg       0.38      0.46      0.40        43\n",
      "weighted avg       0.48      0.51      0.46        43\n",
      "\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [

"C:\\Users\\PRANAY\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sit
e-packages\\sklearn\\metrics\\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.\n",
      "  _warn_prf(average, modifier, msg_start, len(result))\n",

"C:\\Users\\PRANAY\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sit
e-packages\\sklearn\\metrics\\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.\n",
      "  _warn_prf(average, modifier, msg_start, len(result))\n",

"C:\\Users\\PRANAY\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sit
e-packages\\sklearn\\metrics\\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.\n",
      "  _warn_prf(average, modifier, msg_start, len(result))\n"
     ]
    }
   ],
   "source": [
    "X = dst_Data.iloc[:, :-1]\n",
    "y = dst_Data.iloc[:, -1]\n",
    "\n",
    "#splitting the dataset into training and testing datasets\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)\n",
    "#creating a linear SVM model\n",
    "svm = SVC(kernel='linear')\n",
    "#Train a linear SVM model on the training data\n",
    "svm.fit(X_train,y_train)\n",
    "#fitting the training dataset\n",
    "y_pred = svm.predict(X_test)\n",
    "#predicting the target values using the test dataset\n",
    "#evaluate the model on the test dataset\n",
    "print(\"Accuracy: \", accuracy_score(y_test, y_pred)*100)\n",
    "print(\"Classification Report: \", classification_report(y_test,
y_pred))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 144,
```

```
   "id": "e3c54590",
   "metadata": {},
   "outputs": [],
   "source": [
    "#3. Implement Linear Regression using scikit-learn\n",
    "#a) Import the given “Salary_Data.csv”\n",
    "#b) Split the data in train_test partitions, such that 1/3 of the
data is reserved as test subset.\n",
    "#c) Train and predict the model.\n",
    "#d) Calculate the mean_squared error.\n",
    "#e) Visualize both train and test data using scatter plot."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 42,
   "id": "1c9d1278",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "RangeIndex: 30 entries, 0 to 29\n",
      "Data columns (total 2 columns):\n",
      " #   Column           Non-Null Count  Dtype  \n",
      "---  ------           --------------  -----  \n",
      " 0   YearsExperience  30 non-null     float64\n",
      " 1   Salary           30 non-null     float64\n",
      "dtypes: float64(2)\n",
      "memory usage: 612.0 bytes\n"
     ]
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkIAAAGzCAYAAADDgXghAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIW
XMAAA9hAAAPYQGoP6dpAABGXklEQVR4nO3deXxU1f3/8fckIQtLAsRmw0AjIhBCBaRABLWtwa
CUkkfdoGCppWr5ggWpFqxAoC5sVitVQKy/YkW08GgBAU2LoCIaA4ZFwiZoVAQCQsxMWAIhc39
/pBlymQlZmH1ez8djHo/OuWfuPcn3a+bN+dx7jsUwDEMAAAAhKMzXAwAAAPVghAAAAhZBCEA
ABCyCEIAACBkEYQAAEDIIggBAICQRACAAhiyAEAABCFkEIAACELIIQAL/2q1/9St///veb9
Nnp06fLYrG4d0AAggpBCECTWCyWBr3ee+89Xw/VZ1avXq2bbrpJCQkJat68ua666irdddtd
vLa9L5nnrqKa1cudK9gwRCnIW9xgA0xZIlS0zv//GPf2jdGundV69dVXe0DBw5UYmJik69TWVk
pu92uqKioRn/y+vKKjo5t8/aZ6+umn27S0KFD1bx5cx04cEDvvPOrr32
Wi1evLjR52zZsqXuuOOOJn0WgGsEIQBuMW7cOL3/K6dOn1bx5cy+NyjfOnz+v+Ph49
e3bV//973+djh87dkwJCQmNPi9BCHA/SmMAPOZHP/qRMjMzVVhYqBtvvFHNmzfXH//4R0n
SOHGSI+eXdwFdkNh3nJN9xz2QNJNt8sQ1m4x2J+Aza2eEOfGzZwaEODnP/K3dN7mfScxv
sBuF8/zZ9bYVHf8bl9fEKTDbrbKkcJCMiLYPXJi1NqYRBfR3gUL2+w9SOvpQD2KYZ+yFo8Mi
ILatrhFD7+u+cXdQbfWD3V7t/tvgcVnLwcmCYwYZK1t2AJYnj0xUtK4K9QR0tvd7Et/B8UT
ipKp/cVLcgzWWsPu4LdRG6mTPzc9smaY8cC2u7JSgcUbqbfQlPbsYaNT2pY+U9msOpu52wvb/
WxUV7qhyovs6Fb/LRGQFdm58RG+oIQh9yIjwYN9eqU8qQIeUDsmZMFxYv+YjnY87wTV5xXXjoC
YNAOsRTkUvP6Ay9T/l0qfeO7rBh6+oHKDSJdhoMQ8vEY/FtPCwrYUakMVUm8yrb7WBY3z4LGWxu
XSHSQhg3LoS5tLWd4wiBJ6XXkKKaIZzqiDeFTvqhRqD+ePJsyAowb2HzwwdiiywkEz20xlHQ1Hs
LJ1/RdyU1nKKMFwRjUhzrJlmVIRtz1IFBvoZTllp4Ve40ly9FlnN/BFo/BaBGAzMI+CdU2gBh
Dr6tXS97wVsqkdSsP1xxL5OydPqV6pbeAzfvS+gv/2xzDT4HsCc8aHrZBA9R7mn/J5k2A4qmsz
hGBfXz93PDBnbeAkfhUBPAn9W+tspHndfgNhw6MwSJvwOsqCJNy6aJ7cpFvx3bvJMe1u
pUGeDaEtRSk7aZCcE8Nc6xnFLd1nlFu8Fk7FTAYNNUDV55tg2U8EzJJoE+8xvvP2wahG4JIdRo3
YDX40lRbYObYTKZ09v6YWAGEQ1LI9DfLKuZT2v1iEPbgS3zgqEEdLUvwRdzaX2LuaLaIrHH9y
+5sFd9sO2b/qmnSrhqvWaIrs+jprMoybO8ITpVhRrS45qtn6wcXQL9SUdWp6wpgaZ1o9
FLncnY5w2g5bqn5jL8ZnBJSPVxWxD2aG9ws1pPZfGLzozjNzR+xyl9HzeBuWxOH/NS6FR35
B+tXbqx9enOoEIk1WcrbC6E2CAl7SsWcHK2cszwKI8A7wo6IHIAb4MabqxMswNu6YUqrl6z5Q
TLYr4hYyhvnQA0EcT8dTBrqSDDdhybA1vBNNq2PNd8HSJNCaBeQCWO80A0C/Z1g3Em2U9h2NtQB
tlzBZjDpIc+X8EIXz6D+UvJuCgzk3dt2PKIFsycsf/ujOfT0PzvntyrHO0mmTgJMJIDFC7ouUMc
r4MiEgUd0ppyYgKiF9ZBy0oMNNAPUk4Oq9tIt0sqXAuyWMk8mSz1H8Ngw7BXPnO0+1cVF+Bgl39N
e2M3NLnhQ8hmP3VYYQ/xTf6p/sSUBz+qIu0tS95+AP1R6Syo3Cl5F31Q0Ub09V3v67t/ZMMuUjoH
kmvlk1C+MiBfABf6HeqhB1C9kzyz+E1hn/2LeDCZXmC42TVJzMBJxs1wK1H6kbCqJXFgg3kKxM
Wyr5CeXbIlQJ4RO7FdyabYWl7bh1lywzb1HWS28aH5vDRqy1Z6kuWdCvRFRfb3uGa57Emk+RuIFQ3
WhWW8Sk5qS3Yw4fTt6yyxCcEIY0eDAyiDiDWvs8MkRyYCAvJ6XONG3E4Ibpxkg4GGJJgLnhkDmD+E
LuV8I5e1J3ahhZ1b5v3zpBcctr2sJxBldwl5M6XLdVqoKiZfe2+vL9YNO+MtlGk9D17MHEdVeMJf
M8eHSR0KPAAv8ZnO55G3qgvRXT5uvk8lQtQaV6gVN+bDA/RFiFa2tMrZwUGcZfYu3i2lvStp8BbUN
a3rdbSCmzWuQ0Ih9LTPtdV/k7LoVNnB8tOJ1lP2qPJ1pvGxHK7Wd6zVzrpuUPwBzvSMBYG3jfMgY
m8MSYRVzrOMu2z9UTIfkb9rEJSq9mzVLwtjz/5zTKPu2qE2TE5sqIZQn7JrS7f6oHZC7Lj1B3Vm/I
Gvfx9ih/g+7DRl+ZgvdB3aRN6EjOOhbvdtxo1j7iJfpq9lM7Rzc7ec0vF62+wm2uPdeZ8bvh2DrS
ZU2A83M03Bh04rv2onUaaFaJ9A5R/TaS0/3yZqyG4TC99GZu3yxaSuvE5s3cpsNx2ZTRu9cbsRF6x
vG1XAKoH/Y1NWwaHgLiwRwHqs60
```

9VbjySefNAoLC536vfrqq0ZYWJjxwQcfmNoXLlxoSDI+/PBDR1uLFi2MUaNG1XttAA1HaQyAR
0VFRenee+91ao+JiXH87/Lych0/flw33HCDTp8+rb1799Z73rvvvltt2rRxvL/hhhskSV988U
W9n83KyjLNpvzgBz9QbGys47NVVVV65513lJOTo5SUFEe/q6++Wrfeemu955ekGTNmaOnSper
Zs6f+85//6LHHHtN1112nXr16ac+ePY5+y5cvV9euXdWlSxcdP37c8frJT34iSXr33XcbdD0A
TUNpDIBHtWvXTpGRkU7tu3bt0pQpU7RhwwbZbDbTMavVVWu9527dvb3pfE4A94
z127JjOnDmjq6++2qmfq7a6DB8+XMOHD5fNZlNZlNBQYEWL16spUuXasiQIsoqKlJ0dLT279+vPX
v21FlyO3bsWIOvB6DxCEIAPKr2zE+NsrIyIy3XTTTYqNjdWf/vQndezYUdHR0dq6dasmTZoku91
e73nDnDw8NdthsNeBD2cj7bFLGxsRo4cKAGDhyoZs2a6ZVXX1FBQYFuum/eXc8884zL
z6ampnpktACqEYQAeN17772nEzdL7772nEydO9N/rduvPFGR3txcbEPR3VBQkKCoqOjdeDAAadjrtoa
3fv3nrllVd05MgRSdU3PO/Ys/31qZBMgRSdU3PO/YsUM333xzvU+jsUo24H7cIwTA62pZmZGrPwJw7d07z8z58/31ZBMws
PDlZWVpZURV+rw4cOO9gYtAmjQr9dkrpQcgr4PNhlBgcBzAmsnPqtxJFX/+k4ObI
rzbrKTZUUVzGC7HKxbBgPi/P5TrhRD/31aHc7tbVDQKw+z4h4M7m5fhalKUDhJ2YFDwfhS/aUc
6Mzw/oYtcdPXwNgRJhRQjV1P/5LoyN/oRpfsjfvrGj8ZyBQOhK/6kcVwBPWVYwA53fN87Z/vw3
SAAOqtJKKC9WgAF4yMlWz5UrhPpgPqPNFhVhEuyrlW8/rxXjqcEHCjnmyQP4SUCfASZX8ZSW8I
q4Sq47vnfqdkYjHbs8cuwbl7N9vlD02WNU1ICKRTR/p5QA/i7mp4qV2nbmX39dvt4jyNT2fkrh
9TfI1ITX0RPYAx+hI2TmCO0D0KPg2YEyFaEVhWzNDqeBtwc5TADUbU1iwD6mTlxEcaCo+O8ImQ
qUQg8Yb+d6b0Y5C4j+DUsWdaHJ7JVNpDEaJzPkuDiz63+VFKFhWsIqSBv2P8TFQdZ5WUmJ4Xe+
ebTWMUvqffGa9xXwFF7OLDWKJ9I/V5oR7Zt2y8CSoO2H+Lt3lHk0P1kufxhGC4HmDcMgB0BmNvN
5tnWbQZyL1fe/dF7dAPpxLgsd/J3ZNcK7jsxuvOV/XFmn8uPuPjFRpKtHVrz8/bnUdRf2UW/ySM
SMyqXw0jdWSoyZhyVAwvwsiCQtHoYwwEhhCXAVVpR46tO7R0jIpBYIM5Xd76Ie0Snaem/Ybf4IBn
GFOTNBzXqnfmyDIzVeKmd9B8LwSnq1dnwgB/XiEzIS2vcIN6L0UXh3K/FmpYrxvrzU3jY/dRpRl/
Az/6jUNxVnlg/X2YUjl0KK0ZNqwGMwSC6fKG8B6vhTOyR0+yrHlSKrvHl4yK3JrXv4xr47cLfYd
RIJzgd/S1WFtdkxTYSuw1dIh6JB4obf34FYXXyFIhgymG5xbnIEjjnpF2X5c4+S6p7fEPQ/B/5Ep
EtPd6t5upGYyGjqWC2fmIAP4/bNJxepeNkPPPkr6LdeJY2yaX+RVUsHHr9Y5dOvqTsYvjOZjZnVY

4AAL935oz5sXjJMFat8vWo3K4x398WwzBcbfcBVZfS4uLiZLVauXEaABDYPv5YuvjWkNJS6X+
3nASTxnx/s9cYAADBbtw4cwj6+c+r54OCMAQ1FrvPAwAQrE6fllq0MLfl5UnZ2b4Zjx8iCAEA
EIw++EC68UZzm9UqcauHCaUxAACCzejR5hA0cmR1KYwQ5IQZIQAAgsXJk1KrVua2DRukH//YN
+MJAMwIAQAQDNavdw5B5eWEoHoQhAAACHTDh0tZWRfe33dfdSmsZUvfjSlAUBoDACBQWa1S69
bmtg8+kAYM8MlwAhEzQgAABKK333YOQadPE4IaiSAEAECgycmRbrvtwvsHH6wuhcXE+GxIgYr
SGAAAgaK0VIqPN7cVFEh9+vhmPEGAIAQACFhVdkObi0t1rLxCCa2i1SetrcLDLL4elmesWlU9
E1RbRYUUFeWT4QQLghAAICDlFR3RjNW7dcRa4WhLjotW7pB0DcpI9uHIPOCWW6R16y68/8Mfp
NmzfTeeIEIQAgAEnLyiIxqzZKuMi9pLrBUas2SrFozsFRxh6NtvpYQEc9vWrVLPnr4ZTxDiZm
kAQECpshuasXq3UwiS5GibsXq3quyuegSQZcvMISg8XDp7lhDkZgQhAEBA2VxcaiqHXcyQdMR
aoc3Fpd4blDsZRvUj8HfffaEtN1c6f16KjPTduIIUpTEAQEA5Vl53CGpKP79SUiIlX1TS27lT
ysjwzXhCADNCAICAktAq2q39/Marr5pDUMuWUmUlIcjDCEIAgIDSJ62tkuOiVddD8hZVPz3WJ
62tN4fVdIYhXXed9MtfXmh76qnqDVMjKNx4GkEIABBQwsMsyh2SLklOYajmfe6Q9MBYT+ibb6
SwsOonwWrs3Ss9+qjvxhRiCEIAgIAzKCNZC0b2UlKcufyVFBcdOI/O/+1vUmrqhfcJCdU3RHf
u7LsxhSDm3AAAAWlQRrIGpicF3srShiGlp1fP/NR45hnpoYd8N6YQRhACAASs8DCLMjvG19/R
X3z1lfT975vbDhyQOnb0yXBAaQwAAO944QVzCEpLk6qqCEE+xowQAACeZLdXB6CDBy+0zZ8vj
RnjsyHhAoIQAACecuCA1KmTue3LL6UOHXwyHDijNAYAgCdceaU5BKWnV88OEYL8CkEIAAB3qq
yULBbp0KELbS+/LO3aVd0Ov0IQAgDAXVascN4YdfNm6de/9s14UC/uEQIAwB1atpVm73+uhQTY27bvl1au9Ynw0HjURoDAKAp
XM32UAoLOMwIAQDQGKdOOYedu++mFBagCEIAADTU3/5WfVlw4LJRGGMAoCFcz
fYYhvfHAbdiRggAgEuxWp1D0OjRhKAgQRACAKAu8u8+ZJrVub2774orpEhqAwQRACAKAu8+ZJrVub2774orpEhqBAaQwAAFcohYUEZo
QAAKjtxAnnEDRhAiEoSBGEAACoMWuWdMUV5raDB6Vnn/XNeOBxlMYAAJAohYUoZoQAAKHt6FH
nEDRlCiEoRDAjBAAIXVOmSE8+aW4rKZESEy/71FV2Q5uLS3WsvEIJraLVJ62twsNYedrfEIQA
AKHJg6WwvKIjmrF6t45YKxxtyXHRyh2SrkEZyW65BtyD0hgAILQcPOgcgmbOdGsIGrNkqykES
VKJtUJjlmxVXtERt1wH7kEQAgCEjgkTpPbtzW0nTkiTJ7vl9FV2QzNW75arSFXTNmP1blXZuf
/IX1AaAwCEBi88Fba5uNRpJsh0OUlHrBXaXFyqzI7xbr02moYZIQBAcPviC+cQNG+eR54KO1Z
edwhqSj94HjNCAIDg9ZvfSC+/bG4rK5Pi4jxyuYRW0W7tB88jCAEAgpMPFkjsk9ZWyXHRKrFW
uLxPyCIpKa76UXr4B0pjAIDgsnevcwj629+8skBieJhFuUPSJVWHntpq3ucOSWc9IT9CEAIAB
I+775a6djW3nTwpjR7ttSEMykjWgpG9lBRnLn8lxUVrwcherCPkZyiNAQACn2FIYS7+be+jbT
IGZSRrYHoSK0sHAGaEAACBbeNG5xC0dKnP9woLD7Mos2O8hvZop8yO8YQgP8WMEAAAgcLm6Ifr
MGSmap7LQMAQhAEDg8bNSGAIXpTEAQGBZscI5BLlxrzCEFmaEAACBg1IY3IwgBADwf1VVUoSL
ryxmgXCZKI0BAPzbkiXOIeiFFwhBcAtmhAAA/stVKezcOalZM+PBUGJIAQA8D/nz7sOO8wCw
c0ojQEA/MvChc4h6JVXCEHwCGaEAAD+w1Up7Px5KTzc+2NBSGj0jNDGjRs1ZMgpaSkyGKxaO
XKlabjhmFo2rRpSk5OVkxMjLKysrsrR//35Tn9LSUo0YMUKxsbFq3bq1Ro8erZMnT5r6fPrpp7p6v
hhhsUHR2t1NRUzZkzx2ksy5cvV9cuXRQdHa3u3bvrrbbfeavRYAAB+4OxZ1yHIMAhB8Ph6FT
p07p2muv1QsvvOvnGw5cJ5cBQvfcsmoRGBIXxpn3jwtXLhQBlRUUt+dzBTHHm0co0eP1dixY3V
HHjRt1///2O4zabTYMGDVRqaqoyMjIAQMfexR+GuZV3rFQFeUrn4PdcS5CgM3jNV
OcnBwVFRU1aiwAAB+bO9d5YdRNm2gkcKDSrSjzBhwXcQHDnjtAAAQsB1s3rwD2+jmtnstDAZIPaNdvPTMx8yvAwgA/nkWR7IkzhUdQEoihbsBkY+9QEr
ujHGw8I1XlS0NeAGBA5+xrfxc4HIXpXFq9/w2UnJbGt52P6E/SgWTyi79bvEV4JaWCaFLIRB8TDMXIq7Q1dRgcKnjfHsabBuwBLcntb
8Xts/WvXDImRuXj7hQaMXF53tjabsj2ybhpsqTh5uYaBmTs5wVoRvUDVCYpenlxUbdkbmXtO8T3NJ+
wcKGjZJM5WMo2iVV8b6YhpSlpdzkRZyIU8hbUhGKhWqM8bZtHTe7n0cMLuLLEvwwuqLdGWn5v

o2eCbrxR+uADc9v+/dLVV7tvsICXEIQAIISEh1ku7xF5ngpDkKE0BgCo3549ziHo1lsJQQh4z
AgBAC4tI0Patcvc9vXXUmqqb8YDuBFBCABQN0phCHKUxgAAzrZtcw5Bv/gFIQhBhxkhAKhDld
24/CesAlFKinTkiLnt6FHnTVSBIEAQAgAX8oqOaMbq3aYd25PjopU7JL3xa+4EEkphCDGUxgD
gInlFRzRmyVZTCJKkEmuFxizZqryiI3V8MoB99JFzCPrtbwlBCHrMCAFALVV2QzNW75arr39D
1VtSzFi9WwPTk4KnTBYZKVVWmttKS6U2bXwzHsCLmBECgFo2F5c6zQTVZkg6Yq3Q5uJS7w3Kk
ywW5xBkGIQghAyCEADUcqy87hDUlH5+6513nEthkyZRCkPIoTQGALUktIquv1Mj+vklVzdEl5
dLLVt6fyyAjzEjBAC19Elrq+S4aNV1949F1U+P9Ulr681huYdh1P1UGCEIIYogBAC1hIdZlDs
kXZKcwlDN+9wh6T69UbrKbij/8xNatf2Q8j8/oSp7A8pZCxdKYRf9yX/ySUphCHmUxgDgIoMy
krVgZC+ndYSS/GAdoSatb+RqFujkSalFCw+NEggcFsPgnwN1sdlsiouLk9VqVWxsrK+HA8DL/
G1l6Zr1jS7+o10zogUje5nDUFWVFOHi37v82UeQa8z3N6UxAKhDeJhFmR3jNbRHO2V2jPd5Oe
xS6xtJ1esbOcpks2c7h6Dx4wlBwEUojQFAAGjM+kaZV1/h3KGiQoqK8twAgQBFEAKAANCQdYs
iqs67DkHMAgF1ojQGAAGgvnWLJr23WAeezjE35uYSgoB6MCMEAAGgZn2jEmuF031CX87+qfMH
Kitd3ygNwIQZIQAIAK7WN4qqPOs6BBkGIQhoIIIQAASImvWNkuKi9VTe89r3zO3mDs8+SykMa
CT+yQAAAWRQRrIGdU9xPlBV5bxyNIB68V8NAAQKm63uvcIIQUCT8F8OAASCO++U4uLMbf/v/1
EKAy4TpTEA8HeuZoHsdtftABqFGSEA8FcnTtRdCiMEAW5BEAIAf/TjH0tXXLRK9PLlMIAN6M
0BgD+pq5ZIABux4wQAPiLw4cJQYCXEYQAwB907y61a2duy8sjBAEeRmkMAHyNWSDAZ5gRAgB
+eILQhDgYwQhAPCFpCSpY0dz26NhCDAyyiNAYC3MQsE+A1mhADAW3bvJbIZjHV7NAe/ZIXbp4fywA6k
QQAgB3oxQGBAxKYwDgLu+/7xyC2rcnBAF+jBkhAHAEV7NAX34pdejg9aEAaDiCEABcLkphQMC
iNAYATbV2rXMI6t2bEAQEEGaEAKApXM0CHTlSvYYgIBBEAKAxqIUBgQNSmMA0FBFBvvOEcgrKz
CUFAAGNGCAAawtUsUGmp1KaN98cCwCGwUGmp1KaN98cCwG0IQgBwKYYhhbmYPGcWCAgKlMYAoC7TpzuHoF/8ghAEBBhmhACgPQK
BFmhADAFVelsPJyqWVL748FgMcwIwQAkqrshvI/P6FVhV/X/VQYIQgIOgQhACEvr+iIBszeoL
1336uhvS/aEoOnwoCgRmkMQEjLKzqiMUu2qnj2T52OdZn4L/3l3kwN8sG4AHgHM0IAQlaV3dC
TK3a4DEHfn7RGZ5tFacbq3aqyMyMEBCuCEICQ9V3OHfpg2q2mtn91+7G+P2mNJMmQdMRaoc3F
pT4YHQBvoDQGIDRZLLrioqarH16p8+HOfxaPlVd4Z0wAvI4ZIQCh5fRpl0+FfX/SGpchSJISW
kV7elQAfMTtQaiqqkpTp05VWlqaYmJi1LFjRz3++OMyaj11bpuTkZMXExCgrK0v79+
83nae0tFQjRoxQbGysWrdurdGjR+vkyZOmPp9++++qluuOEGRUdHKzU1VXPmzHEaz/Lly9Sxd
FR0ere/fueuutt9z9IwMIFFlZUosWpqbX+9+utP+Vwi5mkZQcF60+aW29MDgAvuD2IDR79mwt
WLBAzz//vPbs2aPZs2drzpw5pw+Ww2AtZ9jdRGXJEHFW7/JVBYYJBnxrWpV8vXSY33jZiIBG
UQUGBWrRooezsbFVUXJh+HjFih
Hbt2qV169Z9pZZo12rhxo+6//37HcDOnPRoPhFuKzfHTjPfzW97g+fjsakrVeOcp/+aVwdE3s/GKxRIvMDvdc/cfOEr01bv/bbp4jqjmfe6jX6+zbOb0ZYq87HHKIYUGbzAGnOKFmmxMzYz68nxgLmf7Az1KzjPcZJ0u6w9Hhgz/0o6+zNtyl9+Xkhc/7VjOEjzCd8R2ar4EKO3L2F3OZcvJUOEJ4GeEbEPdsmFzhm8Gqb+Iq3+i6TzOE3uXePZMfpBLtXuJ9YwN8X0R4MVhGkmM2yBRfuJPpcCIY8pdnD6ZVeajuCzTcYy4Hhk1a5b52IHt9OzdczZrA2LFBu36SY6QhXMZ3h5aWMdBfEKq9QfBxE1x4oXHqSr+wKmlyOo5+pxYgC1y4Zi7Qoe5BETQ8BA8f5Z3dBO3CA7Hnx9XOG2xCP1ixPqkOAh8WRPbzkQbmVk2E6MqsU8sLZ0gQHuS9Ix3OdELJq+xt7FGZMfk7RhUzg8CYYKwPXQU+9KtVJ1+ADkvJUmRKmg02Mw8+F3vAS8MtYHo3u+BNJLKZAHUMI+SgfEy+YVNTS82tzAkwLZcNqZxHVQ1Oa4LaN83uZz7mLwM8C3M57vxnTqQ6gqLqEIZTwIqpIZWlRp5pK4b4mqBpQ2tC2nsMK9Z6jVcyCnWiRvMQzlyA3K0uMt8Lz3b2BkTq7aCbJcCy1CiqFQiuKoCRbpSzLiD2cWl4+KWwaJr9Zkc5GrWAr3lK8cGZ5+LEkiSZkxkP2j4RxkhbQJJ5M2NbM/1EkFyrD3kRxB6FTtfBKgkmzlGy+IOGdwFaQwoi8BPaiQ7BCYSSSCJtSeGtH7e96Ln7SNQ0dLFGWqtJm6dIgw==

EAAhIzQgAazlUprKLC9ewQAAQAZoQA1O/Uqbq3ySAEAQhgBCEAl9axo/PNz0OHUgoDEBQojQG
om6tZoPPnq1eKBoAgwIwQAGelpXWXwghBAIIIQQiAWXS085YYDzxAKQxAUKI0BuACV7NAdrvr
dgAIAswIAZAOH667FEYIAhDECEJAqLNYpHbtzG1TplAKAxASKI0BoayuWaAmqLIb2lxcqmPlF
UpoFa0+aW0VHsZsEgD/RhACQtHnn0tXX+3c3sQQlFd0RDNW79YRa4WjLTkuWrlD0jUoI7mpow
QAj6M0BoQai8U5BD3zzGWFoDFLtppCkCSVWCs0ZslW5RUdaepIAcDjmBECQokbS2FSdTlsxur
dcnUGQ5JF0ozVuzUwPYkyGQC/xIwQEAp27HB7CJKkzcWlTjNBptNLOmKt0Obi0su6DgB4CkEI
CHYWi9Sjh7lt8WK3PBV2rLzuENSUfgDgbZTGgGDmgVmg2hJaRbu1HwB4GzNCQDDatMnjIUiS+
qS1VXJctOq6+8ei6qfH+qS1dbRV2Q3lf35Cq7YfUv7nJ1RlZ70iAL7DjBAQbFwFoJUrpaFD3X
6p8DCLcoeka8ySrbJIppuma0aROyTdcaM0j9kD8DfMCAHBpK5ZIA+EoBqDMpK1YGQvJcWZy19
JcdFaMLKXI+DwmD0Af8SMEBAM3npLGjzYud1L22QMykjWwPSkOleW5jF7AP6KIAQEOlezQO++
K/3oR14dRniYRZkd410ea8xj9nWdAwA8gSAEBDIv3BDtDjxmD8BfcY8QoAB8kun11wMmBEk8Z
g/AfzEjhJAXcE8yuQpAn3wiXXed98fSQDWP2ZdYK1zeJ2RR9c3VtR+zBwBvYEYIIS3gnmSqax
bIj0OQdOExe0lOaw65esweALyFIISQVd+TTFL1k0x+USZ74YWAKoW50tDH7AHAmyiNIWQFzJN
MrgLQ3r1S587eH8tlqu8xewDwNoIQQlZAPMkU4LNArlzqMXsA8DZKYwhZfv0k09NPB2UIAgB/
w4wQQpbfPsnkKgB9/bWUmurdcQBACGBGCCHL755kMoy6Z4EIQQDgEQQhhDS/eZLpsceksIv+c
4yPpxQGAB5GaQwhz+dPMrmaBTp+vDoIAQA8iiAEyEdPMtntUni4czuzQADgNZTGAF+4/37nEN
S9OyEIALyMGSHA21yVwsrLpZYtvT6UKrvB4oYAQhpBCPCWykopMtK53UezQAG32SwAeAClMcA
bcnKcQ9DGT4NQG12SwAeAgzQghqflH6cVUKq6iQoqK8O47/qW+zWYuqN5sdmJ5EmQxA0CMI
IWj5vPRz+rTUoo0Vzu49viA6YzWYBwAs8Uho7dOiQRo4cqfj4eMXExKh79+765JNPHMcNw9C0a
dOUnJysmJgYZWVlaf/+/aZzlJaWasSIEYqNjVXr1q01evRonTx50tTn008/1Q033KDo6Gilpq
Zqzpw5TmNZvny5unTpoujoaHXv3l1vvfWWJ35k+Bmfl36uv945BI0Y4fMQJAXIZrMA4CVuD0L
fffed+vfvr2bNmununtt9/W7ec//W1tt2rRx9ke+Bmfl36uv945BI0Y4fMQJAXIZrMA4CVuD0L
fffed+vfvr2bNmunutt9/W7t279ec//1lt2rRx9JkzZ47mzZunhQsXqqCgQC1atFB2drYqKi78
4R0xYoR27dqldevWac2aNdq4caPuv/9+x3GbzaZbbrlFHTp0UGFhoebOnavp06Z0aJFjj4ff
fSRhg8frGjR2vbtm3KyclRTk6Oiog36P1jw4/UV/qRqks/VXYPhRKLRcrPN7edPy8tWeKZ6z
WSX282CwBeZjEM9/4TdfLkyfrwww/1wQcfuDxuGIZSUlLL0+9//Xg8//LAkyWq1KjExUYsXL9a
wYcO0Z88epaena8uWerdu7ckS8vT7T7fddpu++eYbpaSkaMGCBXrsscdULUlKiyP/dhDp58mSt
XLlSe/fulSTdfffdOnXqlaNasWeO4fr9+/ZdSjRw8tXLiw3p/FZrMpLi50OVqtVbsGxl/V7gffkf
35Cw1/6uN5+r9/Xz72lH6tVat3aud0PZoFqq7IbGjB7Q72bzW6a9BPuEQIQkBrz/e32GaE333
xTvXv31p133wmEhAT17NlTL7300kuN4cXGxSkpKlJWV5+WiwLi4tT3759l+f/f0Xn5+erdevWjh
kSVlZWQoLC1NBQYGjz5022uugIQZKUnZ2tffv26aTRo0en9oV2Xmn25F/9r/X/Onj0rm81meiw
+KT0k5bmHIImTPC7ECT54WazdWWa+zxhRYsWKBPrpP//5j8aNOMuCK0UVvyRJJ
SUlkqTExCeExETT5xITEx3HSkpKEGhtm3bmvq4Okfta/ypz4yjruiiouZrvtf5XamH7zTg3X
OS10s/Fov05ZfmNrtdevZZ95zfA/xms1kA8DG3PzVmt9vVnVVu3dvPfXUU5ZGvaiqSAsXLtS
oUaPcfTm3RzRzevTRRzVx4kTHa5vNRhAKoDu59+MOWJ4njMjqioGHtGm2EBsjnQ9MdBMQOoESAsXLtS
oUaPcfTm3evTRRzVx4kTHa5vNRhqKQH3S2io5Lrre0k+ftLaXd6GjR6WkJ0d2P5wFcsXnm80C
gB9w+4xQcnKy0rpauqel+Dn5ReVPT8eZQTqQoLsmMNM8+ygOyNGGRTpQd+ntV5iBmvXvvj9t9q6mat
j6uzlH7GnX1SXL15SUpKipKsbGxphcCjldKP5GRziHo8Ho8ccDJgTVqNlsdmiPdsrsGE8IAhBy3B
6E+vfvr3379pnaPvGs0Xo0EGSlJaWpqSkJK1fv95x3GaztLZWVqbCw0CwsDJmZmZK0aN7ZWVqbCw0NF
nw4YNstvt6tu3r6PPxo0bdWVVlVZ6eizbt06de7dWUZ1f7C2fGEWmZmpuk6NX1qroPg5dHSj8VSvV1GbYYh
TZnS9HMCAHzDcLPNmzcbERERERxpNPPmns37fe02114zmzZsbS5YscfPPSZNWuW0bp1a2PVqlXGp
59+agwdOtRIS0szysrKHGNGFQdOMjsgaQKZp0mtoQz0NgpN1N29kRpeZg2IvoYaqMdgsMNRUwqyx
L48lUC0Mi4FJZHJudGWAHwCUlgDscZbJBLCyXwrAmeo+hvXDouMSZxY6jIFlW2D0iJ4PCAuA+nK
kRIWpcpr6jVlJsJTT8mNcNxSAtHlVvVusMN7u0fnNLsNNY9FKPBzMvG4ETNWUYXCxUuA+IBLvsz7n
6p0zfjSMsKkrq+TFGcSkw+H1OSnSyYi1nhvRxd+o+n0g1lZaNVdZwt+Ixr0kqcVvwO/EQuBV
TYSxWgxmZ1R4yK0MxYOMHUzwm7rMm2OUx7yaNNVW+Grr5mOzr6zHnDQFDTGuV5zGHEQ1rjd
2FeUWTBXs1dbQZKB2vFEkr+2mXcpbbgQrRwQbxiwvxQHjOMBuiBFU+PNJMTfnzLE51+0vBu+p5
XLnG98c7GRt8FGGq8v+wXsfQhZMsp8TqpP+ufD8PoeqVoA/fnoE1UMhoIx7NVfGhCr7DqGJQ5x
k+PqnCHhNlEAskiOwUh9KMQ6JeF1ENU6xyN+8R/hr3R1t+3//Tw==
Cu0ubjU1O7zTWMBIIQQhBDQ3BYaWreW+vQxt61de1kLJDZ1z7OmBigAQONRGkNAa0xoqHOneT
fOAtXW1D3PfLJpLACEKGaEENAuKzTs3u2xECRd2POsrtu1Laq+j+niPc+8vmksAIQwghACWpN
DQ0KC1K2buW3vXrfuFdbUPc+aGqAAAI1HEEJAa1JosFikb781dzQMqXNnt4+vKXueeWXTWACA
JImVpS+BlaUDQ81TY5JMN03XxARH4CgslHr3Nn+4Z09p61aPL1zYlPOzjhAANE1jvr8JQpdAE
Aoc9YaGsDDnsteXX0odOvh14GBlaQBoPIKQmxCEAkudoeESN0TXtCh02wSACBgNOb7m3uEED
TCwyzK7BivoT3aKbNjvMI/3OQcgrKyHCGIhQsBAKwjhODkahboyBEpKcnx1i1rEAEAAhpBCMG
ngWsDsXAhAIDSGILHf//rHILuvLPOtYFYuBAAwIwQgoOrWaATJ6S2dS86WLMGUYm1wuV9QhZV
r/fDwoUAELyYEUJgM4y6S2GXCEESCxcCAAhCCGR5edXrA9U2ZUqjtsloysrPAIDgQWkMgcnVL
NDJk1KLFo0+1aCMZA1MT2LhQgAIQQQhBBbDcJ4Fqmm/DDVrEAEAQgulMQSOf/3LOQTNmePWHe
MBAKGFGSEEBlelsIoKKSrK+2MBAAQNghD8W1WVFOHi/02ZBQIAuAGlMfivf/zDOQQtWEAIAgC
4DTNC8E+uSmGVla5nhwAAaCJmhOBfKivrvhXsEREAAgPyBADyGf2LD

P7iaBaqqcr1mEAAAbsK3DHyroqLuUhghCADgYXzTwHdmzZJiYsxtK1ZQCgMAeA2lMfiGq1kgu
911OwAAHsKMELzr5Mm6S2GEIACAlxGE4D1//KPUqpW57b//pRQGAPAZSmPwDkphAAA/xIwQPK
uszDnshIdTCgMA+AWCEDxn3DipTRtz26ZN0vnzvhkPAAAXoTQGz6jrhmgAAPwIM0Jwr2PHnEN
QQgIhCADglwhCcJ+RI6XERHNbYaF09KhvxgMAQD0ojcE9KIUBAAIQM0J+qspuKP/zE1q1/ZDy
Pz+hKrufhooTJ5xDUNeuhCAAQEBgRsgP5RUd0YzVu3XEWuFoS46LVu6QdA3KSPbhyC4ya5b06
KPmtq++ktq39814AABoJIKQn8krOqIxS7bq4vmUEmuFxizZqgUje/lHGKIUBgAIApTG/EiV3d
CM1budQpAkR9uM1bt9WyY7etQ5BE2ZQggCAAQkgpAf2VxcaiqHXcyQdMRaoc3Fpd4bVG1Tpkh
JSea2khLp8cd9Mx4AAC4TpTE/cqy87hDUlH5uRSkMABCEmBHyIwmtot3azy0OHnQOQTNnEoIA
AEGBIORH+qS1VXJctOraitSi6qfH+qS19c6AJkxwfgLs+HFp8mTvXB8AAA+jNOZHwsMsyh2Sr
jFLtsoimW6arglHuUPSFR7mhV3bKYUBAEIAM0J+ZlBGshaM7KWkOHP5Kyku2juPzn/xhXMIeu
45QhAAICgxI+SHBmUka2B6kjYXl+pYeYUSWlWXwzw+E/Sb30gvv2xqqir9TptLq3Rs+yHvjQM
AAC8hCPmp8DCLMjvGe++CLkpheTsPa8bCQv9f4RoAgCaiNBbq9u51DkF/+5vydh7WmCVbndY1
qlnhOq/oiBcHCQCAZxCEQtndd1dvkFrbyZOquvfX/r/CNQAAbkBpLBQZhhTmIgP/74bozZ+fa
PAK114t3wEA4GbMCIWaHTucQ9Brr5meCvPrFa4BAHAjZoRCya23Snl55rYzZ6Ro86P6frnCNQ
AAHkAQCgWuSmEtWkgnT7rsXrPCdYm1wuV9QhZVr2vktRWuAQDwEEpjwW7zZucQtGJFnSFIurD
CtSSn7T68vsI1AAAeRBAKZgMGSH37mtvOnpVycur9qM9XuAYAwAsojQUju10KDze3tWsnffNN
o07jsxWuAQDwEoJQsNm4UbrpJnNbXp6Und2k03l9hWsAALyIIBRMuneXiorMbZWVUgT/ZwYAw
BW+IYNBVZVz2ElPl3bt8s14AAAIENwsHej++1/nEPTee14PQVV2Q/mfn9Cq7YeU//kJtt8AAA
QEjwehWbNmyWKxaMKECY62iooKjR07VvHx8WrZsqVuv/12HT161PS5r7/+WoMHD1bz5s2VkJC
gRx55ROfPnzf1ee+999SrVy9FRUXp6quv1uLFi52u/8ILL+j73/++oqOj1bdvX23evNkTP6Zv
pKY63/tTVeV8j5CH5RUd0YDZGzT8pY81/o3tGv7SxxowewMbswIA/J5Hg9CWLVv04osv6gc/+
IGp/aGHHtLq1au1fPlyvf/++zp8+LB+/vOfO45XVVVp8ODBOnfunD766CO98sorWrx4saNm+
boU1xcrMGDB+vHP/6xtm/frgkTJug3v/mN/vOf/zj6/POf/9TEiROVm5urrVu36tprr1V2dra
OHTvmyR+7Xpc9e1JZWb1jfO2nwPr1q3sPMQ/KKzrCLvUAgIBlMQzDIzWMkydPqlevXpo/f76e
eOIJ9eYmXcbeZ1e+6du2q/Px89evXT2+//bZ++tOf6vDhwxw0pMTJQkLVy4UJMmTdK3336ryMhITZo0bxuJvn376oc//KGcvbcaTabUlNTZbVaFRsb65bfS17REc1YvdsUHJL
jopU7JL1ha/OsWuW8DtDHHzuvF+QFVXZDA2ZvqHOD1poVqDdN+gmP3AMAvMZmsykuLq5B398e
mz4YO3asBg8erKysLFN7YWGhKisrTe1dunRR+/btlZ+fL0nTZbVaFRsb65bfS17REc1YvdsUHJL
/e+5OfnO507OzvbcY5z586psLDQ1CcsLExZWVmOPhebOXOm4uLiHK/U1NTL+A04u+zZk9hY5x
Bkt/skBEnS5uLSBu9SDwCAP/JIEHrjjTe0etWzZw50+lYSUmJIiMj1bp1a1N7YmKiSkpKHH1
qh6Ca4zXHLtXHZrPpzJkzOn78uKqqqlz2qTnHxR599FFZrVbH6+DBgw6lYXUmJIiMj1bp1a1N7YmKiSkpKHH1
qh6Ca4zXHLtXHZrPpzJkzOn78uKqqqlz2qTnHxR599FFZrVbH6+DBgw3/oetRZTc0Y/Vul3t3
1bTNWL3bdZmsoqK6FFZefqEtO7u6FGbx3UwLu9QDAAKd24PQwYMHNX78eL322eL322muKjg6s3cmjo
qIUGxtrerlLk2dP3nhDiokxt23b5ryLvA+wSz0AINC5PQgVFhbq2LFj6tWrlIiIhQREaH333
9f8+bNU0REhBITE3Xu3DmDmVlZWZPnf06FElJSVJkpKSkpyeIqt5X1+f2NhYxcTE6IorrlB4eLj
LPjXn8KYmzZ78/OfS8OHmDna71KOH+wZ2GWp2qa9rTsqi6vuf2KUeAOCv3B6Ebr75Zu3cuVPO
t293vHr37q0RI0Y4/nezZs2MxM7dy50/R017p16xQbG6v09
HRHn9rnqOlTc47IyEhdd911pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp
6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
LPjXn8KYmzZ78/OfS8OHmDna71KOH+wZ2GWp2qa9rTsqi6vuf2KUeAOCv3B6Ebr75Zu3cuVPO
ffffv2jRIo0YMcLxv5s1a2YkdvZc47IyEhdd911pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj
nyXWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
LjWrFmjESNGOP53s2bNjMTOnctrKysjcd11pj52u13r16939PGmRs2enDtX/fTXihUXDtx5p89LYRdjl3oAQK
Bz+8rSrVq1UkZGhqmtRYsWio+Pd7SPHj1aEydOEydvZ6tevnyTplltuUXp6uu655x7NmTNHmJdYKl/cJ1Txh1afsK+nq68wH9
+2TrrnGG8NstJpd6i9+Ei6pMU/CAQDgIz7ZYuPZZ59VWFiYbr72Vf2NhYxcTE6IorrlB4eLj

XX6oUXXnB5fM6cOZo3b54WLlyogoICtWjRQtnZ2aqoqPDySN2PdYTQYN9++60SEhL0/vvv68Y
bb/T1cELCyZMn1atXL82fP19PPPGEevToob/85S++HlZImDx5sj788EN98MEHvh5KyPnpT3+q
xMREvfzyy46222+/XTExMVqyZIkPRxYaLBaLVqxYoZycHEnVs0EpKSn6/e9/r4cffliSZLVal
ZiYqMWLF2vYsGE+HO3lY0YIDWa1WiVJbdu29fFIQsfYsWM1ePBgZWVl+XooIefNN99U7969de
eddyohIUE9e/bUSy+95OthhYTrr79e69e9ev12effSZJ2rFjhzZt2qRbb73VxyMLTcXFxSopKTH
9HYqLi1Pfvn2Vn5/vw5G5B7vPo0HsdrsmTJig/v37K2KyMjw9f9Y9/1JYtW/S73/1OkZGRGjVqlK+HF9QmT54sm82mLl26Dw8XFVVXryy
Sc1YsQIXw8tJJWUlEiSEhMTTe2JiYmOY4GMIIQGGTt6e6de8sUHR3t6+GEJLvdrt69dtee01Lly5bptdee01Lly5Vt27dtH37dk2
YMEEpKSn87uF2lMZQr3HjxmxmnNmjV69913deWVV/p6OCGhsLBQx44dU69evRQREaGIiAi9//77
mjdvniIiIlRVeXrIQa95ORkpaenm9q6q6du2qr7/+2kcjCh2PPPKIJk+erGHDhql79+65579
NBDD2nmzJm+HlpISkpKkiQdPXrU0H706FHHs4xMZU1H706FHHsUBGEEKdDMPQuHHjtGLFCm3YsEFpaWm+HlJLIuP
nmm7Vz505505t377d8erdu7dGdGjBih7du7dGjBih7du3Kzw83+U4cOHW0+U4cOHXw0otBx+vRphYWZH
Zv57Cw8Nlt9t9NKLQlpaWpqSkJK1fv97RZrPZtKWZ/p/9Vr97RZrPZVFBQoMzMTB+OzD0ojaFOY8eO1dKlS7Vq1Sq1
atXKUQuOi4tTTEyMj0cX3Fq1auV0L1aLFi0UHx/PPVpe8tlOU8t8BDD+n666/XU089pbvuukubN2/WoWLWidPntSBAwcc74uLi7
kWLtGjRI18PLegNGTJETz75peuaZZ/TrXX//a10MLWidPntSBAwcc74uLi7
V9+3a1bdtWu7N/u6E8gk6lIXA4C4A4e8jvn8e/f3vf/f1ELSTTf
dZIwfP97XwwgpqlevNjIyMoyoqCijS5cuxqJFi3w9pJBgs9mM8ePHG+3btzeio6ONq666ynjs
sceMs2fP+npoQevdd991+fd+1KhRhmYtruN6ZOnWokJiYaUVFRxs0332zs27fPt4N2E9YRA
gAAIYt7hAAAQMgiCAEAgJBFEAIAACGLIAQAAEIWQQgAAIQsghAAAAhZBCEAABCyCEIAACBkEY
QAAEDIIggBAICQRRACAAAh6/8Dkcs4WJELcakAAAAASUVORK5CYII=",
      "text/plain": [
       "<Figure size 640x480 with 1 Axes>"
       ]
      },
      "metadata": {},
      "output_type": "display_data"
     },
     {
      "data": {
       "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkIAAAGzCAYAAADDgXghAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIW
XMAAA9hAAAPYQGoP6dpAABGPUlEQVR4nO3deXhU5d3/8U/WSQjZQLOAgBERZBFZCkZQ24dAsJ
RKtaVNAami8lCsolZxKQS0yuZWVDb9VayAFJ66FBAwAoJADAhhCSCCjUiBABqysCRA5v79MWZ
gmAESmMyZZN6v65rr6tznzsx3iGU+fO/7nBNkjDECAAAIQMFWFwAAAGAVghAAAAhYBCEAABCw
CEIAACBgEYQAAEDAIggBAICARRACAAABiyAEAAACFkEIAAAELIIQgDplzJgxCgoKsroMALUEQ
QiAVwQFBVXp8p8ex0/flxjxozxymt524IFCpISFB9erV0zXXXKP+/ftryZIl/
R6L7zwgj788EPvFgnAKYh7jQHwhlmzZrk8/8c9c//8c//qGsrCy9++67LuM9e/ZUYmLiZb3X3/ryu
vvFKZmZkaM2aHTp0/r9OnTioiIuKz3n3/8cd/1222226264447VK9ePe3Pe3evVuffvqqp
2rdvr5kzZ1b7NevXr69f//rXl/r/dXg+/rdXl/SzAC4u1OoCANQNAwcOdHn+xeP/fecevbsqU8U7
Z0+fVrPPFfecevbsqU8++KFDh3xeE4Cly3eKl/Uk08+qbi4gCLY2kMgPM/ZbeDKPYEZ7Lz8uaBfxZG
EjLvO+/PJLpaen64orrRlBkZKRSUlJ07733+SpK+/fbcWPPPHO0BbcwPTHZ/eT7ZoxJZ/EnctaPpMV4lTDb1N4/H
HH9SHH36otm3bymazqQU2bNh6XXqzRZbL9bsbSimsemjxsEw8ZERERAT77zNoqRNuZ
JxSA8wMmNBqcPAEAZU4gIjOPuJe/W7etIbULt5+ynWd1l/Rq0KL5nBdLXKTF1Q5ysYqVU05Ofw9SQCvpwW9ZCSxAdU6
J8ZOnSoZs6cqXvuuUcPPfV8vZ0vXJ2mvn+/HL+/88wcluMHGCRGl9UDa4Gn+4XU4Q8xUMhyBy0AAjsQeOKBpk83tfQ36LDRCPDsq7QubD0Ak7Hh8d23N8d6oO6Xul1LG5+BG5I3mEN7+KfBj5/3YDXfnGiy32+aG2JKbSHtTB8ENrHyyf53/x5nRs3H9LNP6ZrkEWX2R8dR0b8VTZ6+rK3wHC8k8SJV6ARGFR55XERh3y5Z9CE2lGqNHcmeBG/dqSwHJB+OK/2HV1F2JKvwGPGmvGhF8/p0qvPdQabeXLfIa30Dj2jcZxtp1JwZJ1cAhjAC6mHaK7tOm6ERYCM/wmQlPPY4LXjV5TX+bv41vf1e1HCTndP3qcNcs1lpHrOFU/Mvvn+DXQTXZvdzyGMiOAoSNn72nfkjv2SThi7cQ6Lh1Lq5i+BOkP2oB93nO9/oz5yDLMSXCaJ5pGaWu6dzfn6TEKsGAO7jCiJgHTfPy5axjVsGgevqxetYXSBoTREeFzpBgIrRsPWaBFDZlRZ2YZBExL0dmzSpS2cFUtgjlmjGqBNMuWWb6OdXH1PQEe/rdxaqbVNESi+LrS0tDrUbzlUjL3aXdZsNeRrJyffbJfHpj0fDsA2I03O7bJf7zYYDBVqZ4W/fhmEUZ3xdt9oLzZAdxMFgyTvrZkFqcPpqtD/sNhMWJMEzF3UjaW+vlKI94ucjTuZb3CuE+i/VRA1cWzRUFmKrK7/S3+8+LEMDOQO0e9tk6UPJSHn3J0ZTRehTrvqXsDvo7y9AvNFl8+I+e3ZVtJf/J4AVycNCF+VEq6zKZjp6CwVPASbqz3h8HWavSNm6VKfYDnMKndr9w6QzFRIi8RMpD/evNVUXoy3tEiC82OMEeSxkGGKlRvNRh2DAirp4dPjycvHsqN6Yq6TiT7bUDCBWALzE9hD7dqM1EHIbx1aBABt2rj8Wl51Nd62dmDbP2X2jMbvAhBp8kpkqSDqB/6bRDAN/QZJ4+sB/+UFzCAwhcWvkCXnKRlHULGl/UxzOMg/+gfhh/iwr0h9IAWNBtFHrvc9nf8iBd5MNLfPMH8tPv5C7hM4q6/OiPxFyiKy6/YnA/ARqxStJ8Y3+X2SEJyTJHFhAVsQW1hGS6nBZWaY3nXu0yp9v2m/Xwn3UjcB5pwo6S/dtIHEDiJ+vCyNwKGDY2Kj1w7udgm7Bnw/89UXkeKM4+sFwk9/H/K3NzCY0c0ZmC/+4qbSyxM++DcohDVHBr/eIvTUNmP7YVQLV45AO/2w89R8+ob6Xi0zbhrxlrsZZ3eb4PJOi2tH8BYx73HFK5Jh5TfExT1hsDWdVIQJtodUpqatd2c8X+U9j15xYuuEBcpu6JMR6Ohx8cFXG22bMEkUe8HQ/BjfTLP2LBDD1kRnhJH4Gm3nqPENDsHxBSxnSzr9kPKNQvSjBRJE+5sfDAVGKS8kLdL0ONiIyfKcj7/

/0xRdfaMGCBVq6dKnuvfdevfTSS/riiy9Uv379S3r/kJAQj+Omhq4gEhMTo549e6pnz54KCwv
TO++8o5ycHN12222y2+1q166dXn75ZY8/26RJkxqpCYA7ghAAn2jevLk+/fRTdevWTZGRkRed
f9NNN+mmm27S888/rzlz5mjAgAGaO3eu7rvvvhq5cnRCQoIiIiK0e/dut2Oexqqjc+fOeuedd
3TgwAFJjj+LzZs3q0ePHhf9LFwlG6hZLI0B8In+/furoqJCzz33nNux06dPq6ioSJJO5MgRty
7NjTfeKEnO5bF69epJkvNnvCEkJEkpWn68MMPtX//fuf47t27tXjx4ov+/PHjx62b
Pt2zZUpLjz2Lfvn1688033eaeOHFCx44dcz6Piory6ucE4IqOEACfuO222zR06zFCNGzdOmzZt
Uq9evRQWFqZdu3Zp/vz/T5vEtvf/qZf//rXeueddzRlyhT96lmPVzVVaWqo333xTMEx+vnPf
y5JioyMVOvWrfXPf/5T1113nRo0aKC2bdu24aKioq/9Prrr/ca/ebQmA4E5C3jpBp6R1NCodv
tt27batGnTBX/2+PHjuvnmm3XTTQSLAaQhqk6Ofxt+y2fFMMWxzF/o/zXBEmrvKkGm4gYnErd
v3jz97//+r1asWKFu3bqpoqJCX331lebNm6elS5eqc+fOkqRrTrp008/1csvv6xGXjRJpSVF
Xbt2vazPCeAsVp+2BqBuOvf0+UozZswnnTp1MpGRkGY6Otq0a9f//v3GGGM2btxoM2btxoM
jIyTNOmTY3NZjMJCQnmF7/4hfnyyyy9dXmft2rWmU6dOjJw83OVU+vOdPj98a+Z2Z6Uy
qyZctMhw4dTHh4uGnevLL56623zGOPPPWYiIIu+HlPnTpl3nzzzTdOvXz/TrFkz/Y7PZTL169Uy
HDh3MpEmTTHl5ucv8kydPmgkTJpg2bdoYm81m1uYPjTadOnczYsWNNNcXGxc95XX31lbr3lVhMZ
GWkkcSo94GXcawwALqJfv37atm2bdu3aWwSMyPIAcJYTJ064PN+ciqKUnp
6usrKfHdaHgAAddbChVJ0tOvY4cPS4MHW4IMPHW1HMpzGWQZD744IMLzlm3bp2rZPbs2WOMM
klm/fr1zzuLFi01QUJDZt2+fMcaYKVOmJcdPTVOVn8/79+5s+s+ffq4vFfXXr
l3N0KFDjTHG2O012k5SUZCZNmuQ8Xmxg2m3nvvvfeq9PmKi4uNcxXfyl+QAABAS73Zhu3Y
xxbI12PAYPtroqp+p8f9f4Zuni4mIFBQUp7sdT6LKzsxUF6fOnTs756SlpSk4OFg5OTnOOb
eeqvCw8Odc9LT07Vz504dOXOXLEOSctLc3lvdLT05WdnS1Jys/V0FBgcuc2NhYdeHyYde3a1TnnXOX
5SopKXF5AACAs+TnOzOzEEr1lzZiwnx7EcVgvVaBAqKyvTyJEjlZGROiGRoZiYGElSUQUGDElSQUGBEhISXOaFh
oaqQYMGKigocMbBvS59NdeVFGi0BtqQ9Bv6b0cc5T6tW+e+ecP8L5vTmexqu6Q9AKKkPc9h
cNGkgnT0pdulXO2WsSB06tQp9e/fX2nKHEM7QTJLMqJbUPF3mOK++MD8KR+eMaqX9AAKwhHs34Atr
jQ/Rf/nJmbMoU6YcfpLAw6+rygho5fb4yBO3Zs0fLLy93doMkKSJpKcTYcOHXKZIxf/r0aRUWFsYu
Kck55+DBy5kzp9fbM7+ZxyXHoYrhs8hM7ZxyvHkpOTXebceOnNOHuu22WMyF+Pl5AAKVqHKL25AL
Wd1ztClSFo165d+vTTT9WwYUOXc2WwAsObjLQ7UeGl4tf7PF3ZeWmEJGn+1qEwYEDY
RlZally5aKj493zlm2bJnLa2dlm2bJnLa2dlZSklNVWSlJKSoqSkJc5JSUlysnJcc4JSclVysnJcc4BAAAX8POfu4agO+9
0bI2uIyFUvXPGistLTW5ubkvmNzT5ubkvmNzT5ubkvmNzT5ubkvmNzT5ubkvmNzT540v/zlL81VVq11lNm3aZlL81VV11lNm3aZA4c
OOB8nH0GWO/evU2DhMTk6OWMk6OWMk6OWMk6OWMk6OWMk6OWMk6OWMk6OWMkKBBji8vz8ydO9fUq1fPPT
J8+3TlnzZo1JjQ01Lz44oOWMk6OWMk6OWMk6OWMk6OWMk6OWMk6OWMk6OWMk

C6rzXyyNAQBQR/Rum6xpv22r9I5Xux1bsnW/5+sIBTg6QgAA1BVPP+0Wgna+MVMVFXZC0HnQE
QIAoC7wtOfn1Cm1DOWr/kLoCAEAUJtt2+Y5BBkjEYIuiiAEAEBtFRkptW3rOrZpkyMEoUqIig
AA1DYVFZ67PQSgaqMjBABAbfLSS+4h6JFHCEGXiI4QAAC1hae9QCdOSBHnv6o0LowgBACAv/v
2WyklxX28Gl2gCrvRuvxCHSotU0J0hLqkNFBIMFeXJggBAODPUlIcQehsn38ude9e5ZdYkndA
Yxdsd7k7fXJshDL7tg746wuxRwgAAH9kjGMp7NwQZEy1Q9CwWRtdQpAkFRSXadisjVqSd8ALx
dZeBCEAAPzN229Lwed8RQ8YUO0N0RV2o7ELtsvTT1WOjV2wXRX2wN1ozdIYAAD+xNOG6JISKT
q62i+1Lr/QrRN0NiPpQHGZ1uUXKrV5w2q/fl1ARwgAAH9QUHD+K0RfQgiSpEOl5w9BlzKvLiI
IAQBgtW7dpORzNi0vWnTZ1wZKiK7aafVVnVcXsTQGAICVztcF8oIuKQ2UHBuhguIyj/uEgiQl
xTpOpQ9UdIQAALDCBx+4h6CePb16heiQ4CBl9m0tyRF6zlb5PLNv64C+nhAdIQAAfM1TF+jwY
emKK7z+Vr3bJmvqwI5u1xFK4jpCkghCAAD4TlGRFB/vPl7D9wnr3TZZPVsncWVpD1gaAwDAF+
66yz0EzZ7ts5ulhgQHKbV5Q91xY2OlNm9ICPoRHSEAAKrhku7Z5WkpzG73PA6fIggBAFBF1b5
n17JlUlqa61jbttLWrTVcKaqKpTEAAKqg2vfsCgpyD0F79xKC/AxBCACAi6jWPbuOHz//tYGu
uqomy8QlIAgBAHARVb1n1+GB90hRUa4HX3/dZxuiUX3sEQIA4CKqci+ubyf8wn2wosL9LvLwK
/x2AAC4iAvdi+uGA1+7h6CGDR1dIEKQ36MjBADARZzvnl0eu0A7d0rXXeez2nB5iKoAAFzEuf
fsCq047TkEEGUMIqmUIQgAAVEHlPbvuzl+j3S/2czm2649/ZkN0LcXSGAAAVdS7XSP1Pmesovy
kWoSHWVIPLh8dIQAALmbfPvdrA117rWSMQghBtRpBCACACxk0yP1CiN98I+3aZU098CqWxgAA
8MRul0JC3MfZC1Sn0BECAOBcixa5h6C//50QVAfREQIA4Gye7hNWXi6Fh/u+FtQ4OkIAAEjS4
cPuIahbN0cXiBBUZxGEAAB4+GEpIcF1bPt2afVqa+qBz7A0BgBAIXOe7Hxh7gQIGHSEAQGBaud
I9BL3yCiEowNARAgEnvh4qajIdezYMLmaePUvKgXXoCAEAAkdxsWND9NkhqFAQmSnFxbmOffmltGOHJeXAP7A0BgCo+zxdG4i9QBAdIQBAXfbll+4hKDOTEAQnOkIAgLqp
ZUvp669dx4qLpZgYa+qBXyIIAQDqluPHpago17EGDaQffrCmHvg1lsYAAHXHK6+4h6CVKwlBOCs6QgCAusHThmi73fM48M6gCA2m37dvew8/DDjg3RhCBcBB0hAEDtdfPNUna269jhw9IVV1
hTD2odghAAoPY5eVKy2dzHOS0e1cTSGADAb1TYjbK/+UEfbdqn7G9+UIXdQ7D5+9/dQ9CiRYQ
gXBI6QgAAv7Ak74DGLtiuA8VlzrHk2Ahl9m2t3m2THQOe9vxUVLjfRR6oIv7LAQBYbkneAQ2b
tdElBElSQXGZhs3aqJWLv3APQXff7egCEYJwGegIAQAsVWE3GrtguzwtbBlJb/3rWd02YZ3rg
f/+V2rc2BfloY4jCAEALLUuv9CtEyRJwfYK/WfSHe4/wF4geBH9RACApQ6VuoegPjs+dwtB6y
dMJQTB66odhFatWqW+ffuqUaNGCgoK0ocffuhy3Bij0aNHq1GjRkUpLS9OuXbtc5hQXF+uxxx
5TVFRUVFQU+vr6+pKIqpPG2UIhCq5aGgXyDUZyEOaJkgLWTbY0IV/p7/ZkY8wI0bBVlq72/9jUcVQ1JJgJ73+tcwKPfF+2UqhC9GIrBmHpGPrppSFBeKYePZx1XnpX09
rYtAABrJURHuDz/dsIv9Ma/J7iMXfP4Rzp91298WRYCRLWD0LFjx9S+fXu98cYbHo9Pn
DhRkydP4nhzZR7P3bkpUyCzBRqFb1aU3lEGrXWxklvXmUWZbQdrJSGSdeF4ReF9rYzkTrk
qpWbNm2rBhgyZNmqQxY8ZoxowZjrlXwWSg5aUoocfQ2+cPdS7OD7oi5KZrfKZCQ2mWaHdOfzEOrZy65TuZMBxeAHyLsmjCeNMnCfgFVV8pfBSeehZbz7K64MjU69QeXcfuPeB
eDnKbv+KVzQVgzRgbp8ykDxO1A4Cr
DJcdGqElRgb6d8AuXY582/4lSRi5UYnyUuqQ0sKhC1GnmMkgyH3zwgfO53W43SUlJSc6xKkSc6x
oqIiY7PZzHvvvvWeMMWb79u1Gklm/fr1zzuLFi01QUJD5+NFWPy3DsvUW7V4Gq+fMcaYKVOmmPj4eFNeXu6cM3kSS
NOyZUvn8/79+v5s+ffq41NO1a1czdOjQ5KtdyMcXFxcFlv4yuTvfIBGoqHcoqi5OoW6bsMrVdiVr0Dvcs
iryY40V49caBZv3W91iahFqvP97dU9Qvn5+SooKFBaWppzLDY2VPt6dlX27DzA0Xz7Dz8VlxcnDp
37uyck5aWpuDgYI/3rfPDvSOXXyg9Dj3pkZ7PS6zOrKEgKgG0IGlq6nFSPWXyBnMLPH4eVn9UpfpDzsTxdp3JcQ3
vuzS6cl+o9Wszl4mxGNhnmjUPZ8lmz/YeTxZsySuGkWu
a3gnlwfmA/f9RU5f3ZV7jEo0jAeYuVy2u0WzA+sezdd8odQ4GQr9HnzcZ/lvJ/Qf7/f8jlbIlBBmmwdvemlsVe5yNbtZ7ZPH2B2f+rq4Y+dDO+cmfWGZe9mHwu8fRfPIWYVWyd3DfZ
6WMp1bfpFq/L/2+fWMOGHy/+ZsP1lFD7Esx3ZnbBsOztyVuKbfdPkcsKp3QF1Hs8yvQ2EHQszLq0dFWJt3cFjyn/N8/bWLYz8vaxAZ4Pl2/FR4CqjDFdEbNIsKDCsycNNH8Z5B6wt1nTP8bQFAIb0D5SVCBsDQp9tdNkc/XOQGyhfZ
Ow1v3yOwe7IrMFl9fKSPRPnkNDNO7BDD6VjPfjlKIk5X2T62YjbU1HeBoFeoKLYEKGrtbvrc8N7aiVrbJTYNKeS5/CS5zkQsH+wsBJdM5f2xK0hdUN2P4yB2xCUs/jWrh/9lPzq9I9u9Y6Rv1jtAVH9B9IFeWlO0HTbTyxjOyMMmzf56rQOUqwFvf2ZI8dTUbUbrA+yeF7xRb1kJ0hQkoF8U7yBMjrLhyMvg1h8Mq5bMw91qSCJoOALXlYjG2vD9Q0lmSRo2YRR4F4KHpSjbL3DRyNA9Sf3A/4NtCKrRC5vN0nU==

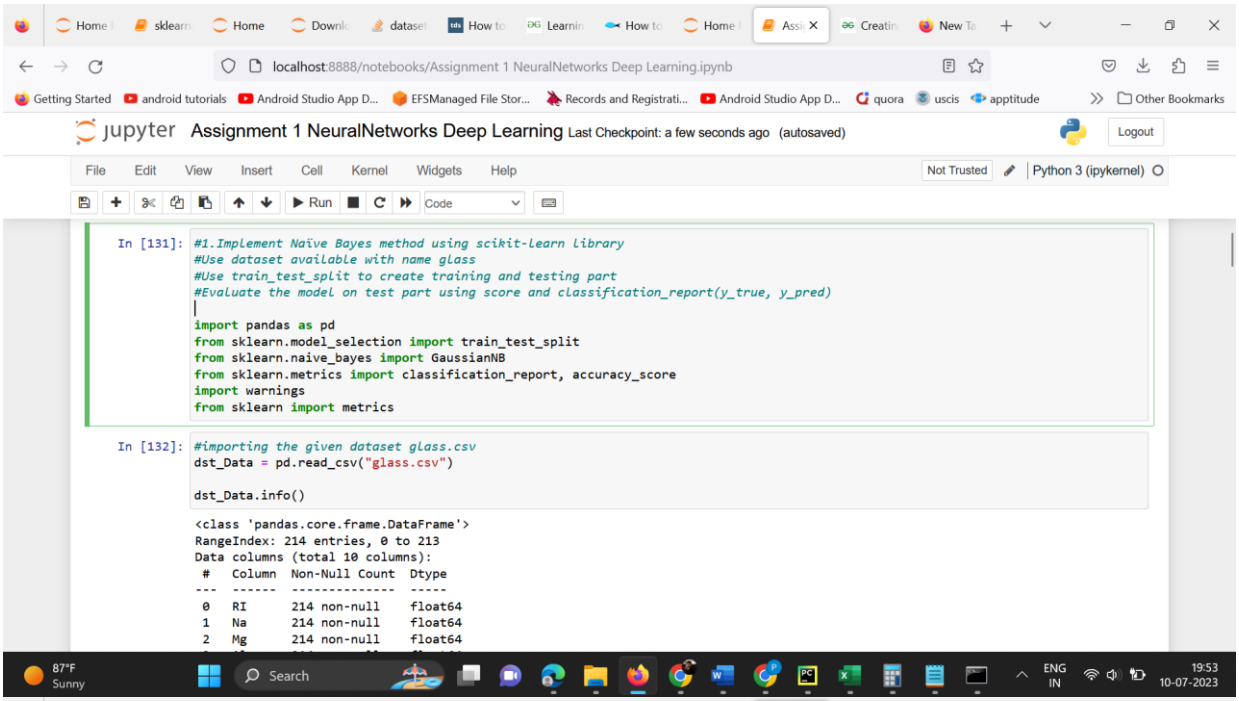
GsLJbCgEvA0hgA1Cbn6wIBuCR0hACgNvj8c/cQlJFBCAIuEx0hAPB3nrpAhw9LV1zh+1qAOoY
gBAD+6tgxqX5993G6QIDXsDQGAP7oqafcQ9C//kUIAryMjhAA+BtPS2F2u+dxAJeFjhAA+Ivc
XPew07OnowtECAJqBB0hAPAHDRpIR464ju3dK111lTX1AAGCIAQAVjp5UrLZ3MfZCwT4BEtjA
GCViRPdQ9Df/04IAnyIjhAAWMHTnp+KCve7yAOoUfw/DgB86auv3ENQ+/aOLhAhCPA5OkIA4C
utWkk7d7qO7dolXXutNfUAIAgBQI2rqJBCPfx1y14gwHL0YQGgJs2Y4R6CXn6ZEAT4CTpCAFB
TPG2IPnlSCgvzfS0APKIjBADe9t137iEoMdHRBSIEAX6FIAQA3vTTn0rNmrmObd4sFRRYUg6A
C2NpDAC84Xynv7MXCPBrdIQA4HLNm+cegkaNIgQBtQAdIQC4HJ42RB8/LkVG+r4WANVGRwgAL
sWhQ55DkDGEIKAWIQgBQHX95jeOs8DOtmYNS2FALcTSGABUx/m6QABqJTpCAFAVS5a4h6A//p
EQBNRydIQAWKbCbrQuv1CHSsuUEB2hLikNFBLsoeNiNU9doKIiKTbW56UA8C6CEABLLMk7oLE
LtutAcZlzLDk2Qpl9W6t322QLKztLcbEUF+c+ThcIqDNYGgPgc0vyDmjYrI0uIUiSCorLNGzW
Ri3JO2BRZWf54x/dQ9DSpYQgoI6hIwTApyrsRmMXbJenOGEkBUkau2C7erZOsm6ZjA3RQMCgI
wTAp9blF7p1gs5mJB0oLtO6/ELfFVVpzRr3ENS/PyEIqMPoCAHwqUOl5w9BlzLPazx1gQ4dkq
680rd1APApghAAn0qIjvDqvMt2/LgUFeU+ThcICAgsjQHwqS4pDZQcG6Hz7f4JkuPssS4pDWq
+mGeecQ9B8+cTgoAAQkcIgE+FBAcps29rDZu1UUGSy6bpynCU2bd1zW+U9rQUZrd7HgdQZ9ER
AuBzvdsma+rAjkqKdV3+SoqN0NSBHWv2OkKbNrmHnZ/9zNEFIgQBAYeOEABL9G6brJ6tk3x7Z
ekrr5S+/9517LvvpCZNau49Afg1ghAAy4QEBym1ecOaf6OTJyWbzX2cvUBAwGNpDEDd9tJL7i
HorbcIQQAk1UAQqqio0KhRo5SSkqLIyEg1b95czz33nMxZf+kYYzR69GglJycrMjJSaWlp2rV
rl8vrFBYWasCAAYqJiVFcXJyGDBmio0ePuszZsmWLbrnlFkVERKhJkyaaOHGiWz3z589Xq1at
FBERoXbt2unjz/29kcG4K+CgqQ//9l17PRpacgQa+oB4He8HoQmTJigqVOn6vXXX9eOHTs0Y
cIETZw4Ua+99ppzzsSJRNmzZNOTk5ioqKUnp6usrKzlxAbcCAAdq2bZuysrK0cOFCrV
q1Sg888I8Ee38IDzeElJiXr16qVmzZppw4YNmjRpksaMGaMZM2Y456xdu1YZGRkaMmSIcnNz1a9fP/X
r1095eXne/tgA/MnXX7tvfG7TxtEFCgmxpiYA/sl4WZ8+fcy9997rr7MnbnnXeaAQMGGGOMsdvt
JikpyUyaNMl5vKioyNhsNvPee+8ZY4zZvn27kWTr1/vnLN48WITFBRk9u3bZ4wxZsqUKSY+P
t6Ul5c754wcOdK0bNnS+bx+0aWOMI/KceezcaXVVAHyoOt/fXu8I3XzzvVq2bJm+++07S
zsbGx6tq1q7KzsyVJ2dnZiouLU+fOnZ1z0tLSFBwcrJycHOecW2+9VeHh4c4ED3rIc+N3zPPcdJcpfBQj
OeKcc/b7VM6pfJ9zlZeAq6SkxOUHBoJaoqHB0gbZtcx03RrruOmtqAuD3vB6EnnzSf3f79fl/59Tq
1atFBYWTpw4QYwZM5xWWRkmzpwQ3vAnA73mQ6EsuOX5vO+QXJIPhgNDknatGqzu0nk+zcgrk3vA4
e/x/nmVB4/17hx4xQbG+t8NOGUWqB2+PvfpdBzToKdNIkN0QAuyuunz8+bN0+zZf8/bN4XFqKZBehZeR1jTlwneeh78rQ+7U1JOFM4n/Dg9Xv8z969UtOmrmNXXCEDdPmXNmbPQc
8vjR0/flzBwa4vERIerpKAllRlRlUlKSli1b5jxeUlKinJwcpaUTDAgJWccpaamSpJSJJU1NVVFSkDRs2OOc
sX75cdrtdXbt2dc5ZtWqVc3Hk4bk14dj4enjn7PepnFp5PgBqqR493ENQbi4hhCEAD1
eXun9uDBg03jxo3NwoULzcqVKw3SZTqXEpvtDc2UdkcXlgEudTnepLHmG1nbRn1rfy/XYvT8k9Qyvk
KSkmBMnTjjn9O7d23To0Hk5OSY1atXmxYtWpiMGdhgDRpk8vLyzNy5c029ev
XM9OnTnXPWrFljQkNDzY+WvvmH27NhhMjMzTVhYmNm6dWuVPgeB7hnjQFS+xm53PyPM+3+NAajlqvP
97fW/QUpKSszcDD9smjZtaiIiIszVV19tn+dnYGDkRJTEw0NpvN9O9OjRw9z64jLq/a6sjBb0nVApEnSQAJQE
H374wWRkZJJgYc88emu68NzxVhnAenDezsynmZNNqqhmZrNnVrzKa2Sqnksj1k0M8vmcwyGVqmnZCV5ozyvG8tI7qZs/Ghog8EOG4nQ+jqKxMzvdmUbpPS
MCKr2YsUhgS6tQVQL9tUx0W4A4A04dHUqcOmf3Ai0J5rPvsjZtq6XF1Zw2AkHV3ejZAGG0ujcb2xcI2uDCIABmuDIc6WXCo/tPcudzWuq47KMqvt6AmFDe
nlxLXx8UBdS2Bj7QhoIFUcmqoU/GCaM6aKK+tYMkCxp+4Fw8mdP7bNh2O2ss5zdbdsZKj8XSWMDEXyX7zmXiGy
/m2Hsn1lGq+fY/bTHhrrZOMUsztmbzvyGNQBNGkSbbXQ//u6zxGZM3Pj8Pep9bkwyENOGDLrXQ71Ph+7njQpqmoBzKmNSVMMBp8Cn8ZcvdEI8a7XBA8QCPwCNWfA8Lvvwa+3nsNjpA9eB+3OpdN1kmRdEHTRy4NzZ6JvfNl1xEEbnZzRvs3mA3Tmr2uWNNbaGSLzOnzbVhdcTrqQzZqJ6gE0DKZ2YJK9wFFiTUAhQmGhzXA83u5AeCw9Y6MBgqpYrazMX3+pF+zWdYdOdQ6T2oK/SUY4N6
AjBKDmsCEagJ+jIwTA+zZvdg87t9zChmgAfoeOEADvSk6WCgpcx/bskZgAghAA7zh
1SgoPdx9nLxAAP8bSGIDL9+qr7iFo+nRCEAC/R0cIwOXxtOfn9GkpJMT3tQBANdERAnBpdu92
D0GtWjm6QIQgALUEQQhA9d14o9SoOdNNQZtp65i6U91zt4e9QABqKTpCAKpm5kz3
DR+PCEIQK1GRwjAxXnwHbo9GtF5Bs0SVMOBFy0jOU8kNHuDVTBDoAG2eAhgrCAA2tSoCYW2fV2etMHnw5JC1AOE1iQ3GPrxMeduMx1J1V4w6VCqAuTaZYYkm9yOAUKu1ISZabpqTHUprmZfAF0zqYp5ZN1VE6zH
bKCQA5c1Bh1cB5z9kIoXQnq4DlsxC1Hn8wd5H8QA6vMsmQ2LbbWmK7vJQAXjqBQnAQ5AD8QoAq7Zj3ZzWGEbjzd5RK6J3bvuPL0FNvXhiGfmzMeF3ZQzFMKQvVuJW7/oc6N2GH7C3Yrpx
2KEewhauJAQBACiIwTUbZ6Wwux2z+MAEIDoCAF10RdfuIedfv0cXSBCEAA40REC6pqwMOn0ad
exAwekpCRr6gEAP0YQAuqKsjIpMtJ9nL1AAHBeLI0BdcHYse4haM4cQhAA
yOkJAbZWX5x52br6ZDdEAUA10hIDa6KqrpH37XMfy86Wrr7akHACorQhCQG1y6pQUHu4+zl4
ALgkLI0BtcXkye4haOZMQhAAXAY6QkBt4GnPz6lTZ/lTUij/FwaAy0FHCPBn33MTH+
cBlIwgB/qpzZ0foOdv27dKuXdbUAwB1EP+kBByPyHS6hLiPsxcIALyOjhDgT+66yz0EjRhBCAKAGkJHCPAHZWWefxvs2DHPnkHAAOA10gAaggdIcBuFeqR+/RxdIEIQANQYOkKAVXbtklq2dB/fulVq29b39QBAACEIAVawx+T3hQuLOptbUAwABhKUxwJc2bPAcgjwdkwEAqDF0hABf8RSAsrOlbt18XwsABDg6QkBN+4//8ByCjCEEAYBF6AgBNaW8XIqIcB8/eFBKSvJ9PQAAJzpCQE24/373EHT4sKMLRAgCAMsRhABv2rHDsQw2c6b7sYYNfV8PAMAjghDgLcOHSzfc4D7O2WAA4HcIQsDlWrzY0QV67TXX8ePHCUEA4KfYLA1cqiNH3He+ePNxVkOLu9tb8UJuT9yxXfADsoL8i+6bKGyL/tG/8VHpKc7j3aXAjKCoCtRWnLCYz2eEWvK+VpC2IMTK1ZKwAHORO7k6H/y1P+S6/0K4Ab5SjQw+0zUpr7y0PIBwTmBmAVGz6XUTQ6Dx3KtSFdZHIcVQeSdKUQ1k6ZKdm0TZmq4ZNKgEyl/g3IvGvAMgK1bLDvP57Ko1j5rLLM6AdGR0QUo4Zql7M1QtO6vBZYOy5bOQ7zMfqDsmqgDzeRmj3tW3ILCmn4HSTN05pdAScSNYJdoGItpkYiyjIS4o3YXDKPgx5NyJ8GUQxe2S3Jwv
gAaggdIcBfeNoQXVYm2Wy+rwUAAgQdIcBq+/e7h6D69R1dIEIQANQoghBgpZ//XGrc2HVs/Xq
ptNSaegAgwLA0BljBGCnYw79D2AsEAD5FRwioQRV2o+xvftBHm/Yp+5sfVGE30qZN7iHoscccI

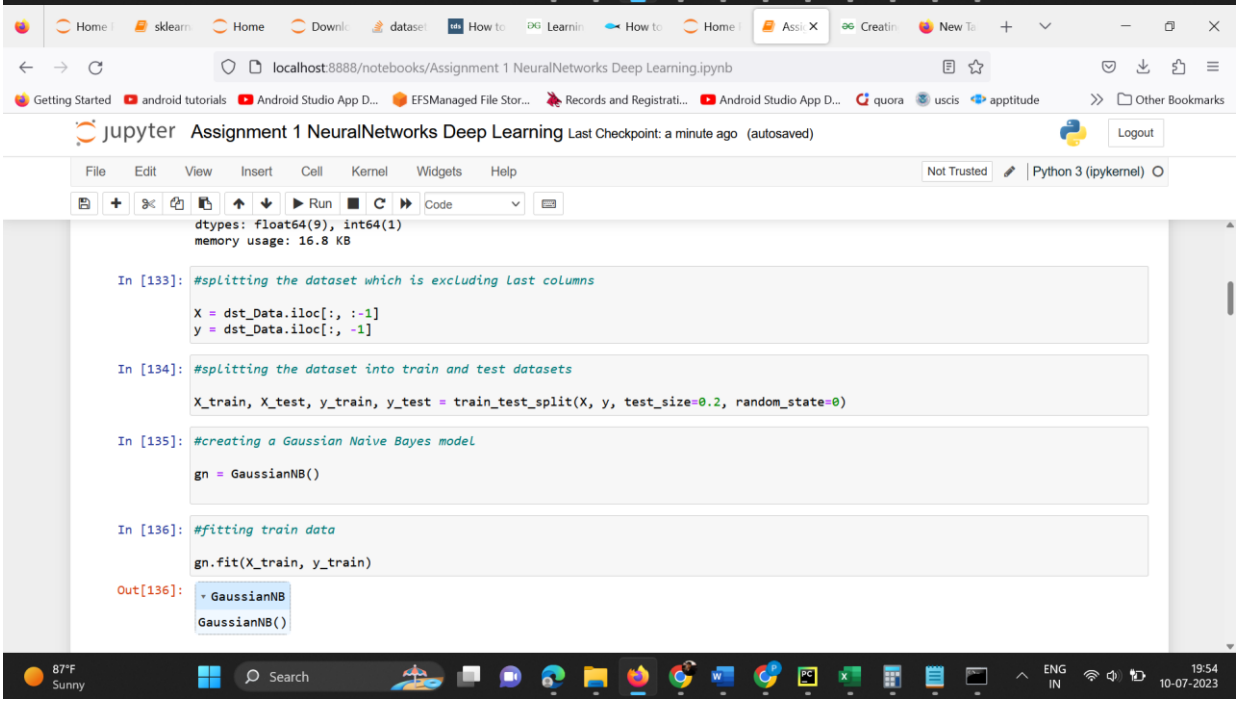
QQBgATpCQA1ZkndAYxds14HiMufYrI/+qu5ffeE6sbTUsScIAOBzBCGgBizJO6BhszaqsscTe
bJMO175teuk1FRp7Vqf1wYAOIOlMcDLKuxGYxdsd4ag32zJcgtBv33kbVWsXuP74gAALghCgJ
etyy90LodlbFqiSYv/5nL86pELlRN+pdblF1pRHgDgLCyNAV52qLRM9cuPa2zWVN21bYVz/M8
/H6H/a5fmMg8AYC2CEOBl13yzTYtmPqRmRQWqCArW5Jt/p9dv/q0qgl1vnZEQHWFRhQCASgQh
wFsqKqQJE9R29GgFVVTovzEJerjvn7XhqtYu04IkJcVGqEtKA2vqBAA4EYQAb9i7Vxo0SFq5U
kGSDtx+h35+XYZKI1xPi6+8kUZm39YKCfZwbzEAgE+xWRq4XP/6l9S+vbRypeN6QO+8o+RFH2
jifbcqKdZ1+SspNkJTB3ZU77bJFhULADgbHSHgUh07Jo0YIb31luP5T34izZkjXXutJKl322T
1bJ2kdfmFOlRapoRox3IYnSAA8B8EIeBSbNwoZWRIX3/tuHP8k09KY8dKYWEu00KCg5TavKFF
RQIALoYgBFSH3S69/LL09NPSqVOOO8e/+670s59ZXRkA4BIQhICqOnBAGjxYyspyPL/zTmnGD
KkhHR8AqK1qZLP0vn37NHDgQDVs2FCRkZFq166dvvzyS+dxY4xGjx6t5ORBLTbt27X
J5jcLCQg0YMEAxMTGKi4vTkCFDdPToUZc5W7Zs0S233KKIiAg1adJEEydOdKtl/vz5atWqlSI
iItSuXTt9/PHHNfGRUdctWCDdcIMjBEVGOgLQ//0fIQgAajmvB6EjR46oW7duCgsL0+LFi7V9
+3a99NJLio+Pd86ZOHGiJk+erGnTpikJ0dRUVFKT09XWdmZK+0OGDBA27ZtU1ZWlhYuXKhVq
1bpgQcecB4vKSlRr1691Rr169KxZM23YsEGTJk3SmDFjNGPGDOectWvXvXKiMjQ0OGDFFubq769eunfv
36KS8vz9sfG3XViRPSgw9Kv/yl9P330/Y28QAKB2M142cuRI07179/Met9K
pyUyaNMk5VlRUZGw2m3nvvfeMMcZs377dSDLr1693zlm8eLEJCgoy+/btM8YYYM2XKFBMfH2/K
y8td3rtly5bO5bNsZIjsdjj
xlTVmZ1VQCAi6jO97fXO0L//ve/1blzZ/3mN79RQkKCOnTooDfffNM5vPZ8Z8PP/XwUFBUpLO3PPpd
jYWHXt2lXZ2dmSpOzsbMXFxalz587OWlpaQoODlZOTo5zzq233qrw8HDnnPT0dOTn5+vIkSPO
mlZ/DlZOTo5zzq233qrw8HDnnPT0dO3cuVNHjhx
xzjn7fSrnVL7Py1VSUuLyQAAyRnrtNcfp8Nu2SYmYmJ0tKl0osvSjab1dUBALzI60HoP//5
j6ZOnaoWLVpo6dKlGjZsmB566CG98847qSCggJJUmJiosvPJSYmOo8VFBQoISHB5XhoaKgaN
GjgMsfTa5z9HuebU3U3n8KePtGWL1KuX1Z
UBAGqA14OQ3W5Xx44d9cILL6hDhw564IEHdP/992vatGnefiuive+qpp1RcXOx87N271+qS4Et
Llzo2RH/8saPz89prjk3S54RyAEDd4fUglJycrNatXW8yef311311JUmSDh486DLn
4MGDzmNJSUk6dOiy/HTp0+rsLDQZY6n1zj7Pc43p+rsLDQZY6n1zj7Pc43p/L4uWw2m2JiYlweCADl5dJjj0m9e0sHD
0pt2kjr1zs2SbMhGgDqNK8HoW7dumnnzp0uY19//bWaNWsmSUpKSlJSUpKSJSVFSUpKWLVvmPF5SUqKcnBynVEcnB
ylpqZKklJTU1VUVKQNGzY45yxfvlx2u11du3Z1zlm1apXKysqcc7KysnTdddc5r1BLTU11e59z39XvRBgVu8T41kgGiIZ9iaUgL6o1BAbgWEwlqzr+ROCw3qqQCigJ+N/HD4h46M9fuN2U01nsAV4kvh35HJpICLJuYaN+DVufzgUcnFlQ
KOZXvA2jHDummmxwXSZQc4Wf9eqldO2vrAgD4hrd3aq9bt86Ehoaa559/3/3uzatcvMnj3b1KtX
z8yaNcs5Z/z48SYuLs589NFHZsuWLeaO+4wKSkp5sSJE845vXv3Nh06dDA5OTlm9erVzdDZ6D4b+bf8cVP3ntlVV5P6PhqDsdsrwM2QNXbPDfwfi91udFaO5/LZVNK2G+nqaqSJmsENSH1YPbk3tSNXdF5XCSN4dxSJUtcCndAC+
iYjI8N5vKioyCQmJppBgwaZvLw8M3fuXFOvXj0zfeZpgpq3OLUiE+5o6tNjd9SvQuw1KOoOz2/KinvOjvcQa1dq4WcSGfk6JX7XjXFtWgcVsBw/bqXVeYnbPodZFTcsxCMxQdBwd6MHO2jUaMOmrKIi5qKFlYMXcfJdbomWqwdBeUhlJ3M3/DcVq6XtxS3bcWSyWZg8LNVMGG2LNkuQdpKiN9KSQ5nPLspCIzJ0gs0vgDgmgN/Ui/eamg0uSi1eo+QU3WqL2sErtsHMGCRG3jx+5wk2glHb8Gt7Zb3mq9Q0Lj2N02BZ3KFEW1FgDrqHXMvRxRuv8MZpIPEnwQdGrnG5Nep20aRR1Fi/UFhBkRb9VY3Yq7WK1UVFnsq4t3Yz42O7UaOB+MLpjlmOuAXGhRoDdr9XGrSx6EGVLd+mZCY9nAXwWUUqsKSJNPUoIa9+oimfVhYqKSbqHKpu0+4BXFuf80d0H0Gm6yJpP0IkyE9XnpPCcCxbKNZXWnzDBfNiyeW0W0TyPj+t1kIqYw1oFbd/cWUGj6MT9NpiPvuNdhJyIpAQFYPKNMYDNEJE8Va3b6BJ0SibXlbDlFXXqEYCkn4JvdcUU77JPGjU9Ltg1O0aUuCXMOKoP8TKpyO1Xlc4SBaMVCZpH/UQL6vE3JTp1/rljUAqvdMVoFXXXSJUHNpNt/D7fUSJKB+7p2XCKGs7tDBmUmgTVKFKwwVJQJGbFnVEe4dLfUAMvzWcqP4CcmtN91fQZOtBXmbNvWFGADAJATVL+7yKK4uCV5ZGlNm1r+mvzbsLeTdL5eqA5/S5fmGyb2rBkpTfQeDBPwBYDvtuZsMiBKHF4TlqmBxNxJQQLnoLTAB+6sJoGDYUrk3hHwlPuiE+3MqKmLu5M7l+BPbV1ZCnZBGiYMJRfHDCD46PYqzsyc/DzGBLMZ31QCRrjdQmTUm+4q2lVtIDGP9qlH69r0kNqHkSDS5cxVsWnRMwBPQcGNlD2VGQjXKc4Aw2+LnOKmoKfSGEEqWLWcKC6Yb0g3h0HDqEYd/pOLzZLZZfJfNWdDiPXZRhBI4lX7yVQpsjvnfBa4AHEYeqK90nBQkM7xgOtAyYUHTSY54X7DnvJVvG6ftyJTxUgZPG7FAdGHzP0yEvSaeWrHCc0/nrr6FOpV1lGwDRGVkRY5t2FUTZGXc3xGzhJaUwOjKQGZdGDJGMMvGqd7kJ2EE3ZdMC04Xje1oRsXhzuFg+8N6WYMmU1E4/w5Pm/+DQ9GYWBoLgNJWVGBxe0WxGsAaG0VO/JZzdqbnxG9WWQpYOLt8aeyV48uQgwq0aQajY2qxCVtRJfMeY8MxaJTW9xgXW5yqLKeBgRT1oHz9d7xkyS1ZpeKxXaXzaqJU7SS7hYbS1kYFbgi0cXJHIMQ3BijIP9uWbr9aCW6OJaxzEjc4E2+CFqlbN9OYrl1JRtHR4vlkPjhVEWnfrOj48tN3iJA/ZYXVe9PHHj+5lDsqLTFMdUPlLlmqbvDAAA0o9D5U0qzzK2M8Y5XKdYf4z9sX2Y1A6jBW2KCKUjfCMAIMeV+4I8dDaEKMG1Fmhqn3zZXzkDlZQqMFzTzdPXjhJPl4Mmz1q4TXYzYSjBw3t7fjkSGmeuKp1nbMqrzEmyhfVS5UBB2akQBKLZhXvBdJv9qIRSJPV3oxrZbyYw8OTmlydl4tQpsd3Q2IcbhqO9nDevMb1OuKXRg4g4u2GYvn/XdbB1KdJWfrqZZMsmzZFYAnJ0AMMRYwBEfKNmi9zrFGzZMYwQ1eRdRLFfMSE2L5vWGk/zZ4q0M0AvSOrqBl9jkSCQo1bjQXRNlxRIrIY4JENLbIB7Sjk4OJPQcQSQIY9RvkBXbbXlZp1QbKnLEOLCNHbx+Y+4tqFhE6x9u3QxAJbUYMQfGgG8L9MHj0jOKq0LDyWWcSRs5TMpSvjK9pPCE2g+5eI2yqPoAAF2BZUzrqkt7CwNApU8nYnNgXAh6q9Ux/QZLJAKxyy+zJ0UmjAB9i5NO4TLq7UWeaoSXAHNdaxc+5p0hq9LAgKVzqM6i9pmHVMGHu9HrfaNAVLj8cZ8Mg/6Ymzcbc8cWR8HgdPyhBnLkpS9MeEYRV8cZ7QkCnKGz+Do4WA0aBDTLnpDLZgBhzMWAd/V3a9IxGS9yKCpgTdS8H+UGqNlD3ga/QvbsnsedpP2SpxjPoGZm/dAHdo/
ap48ZJjzwiBdfIRdarrMJuuEs9AHhJdb6/udcYAsOpU9Lo0dKECY4+UMuW0pw5UseOVlemJXk
HNHbBdh0oPnNl9eTYCGX2ba3ebZMtrAwA6j5r/xkM+MLu3VK3btL48Y4QdP/90oYNfhOChs3a
6BKCJKmguEzDZm3UkrwDFlUGAIGBIIS6yxjpnXekDh0cZ4LFxztulDpjhhQVZXV1qrAbjV2wX
Z7WpivHxi7Yrgo7q9cAUFMIQqibioqk3/9e+sMfpKNHpdtukzZvlu66y+rKnNblF7p1gs5mJB
0oLtO6/ELfFQUAAYYghLpn9WrHXeLnzpVCQqTnn5eWLZP85JYpFXaj7G9+0OIqLnsdKj1/WAI
AXB42S6PuOH1a+utfpeeek+x26ZprHBuif7wIpz/wtDH6YhKiI2qwIgAIbAQh1A3ffisNGCCt
Xet4fvfdjnuF+dFlDyo3Rld1x0+QpKRYx6n0AICawdIYar+5c6X27kX2ixsnhR2EGmj3bsUnaj0LQh
TZGe1J5BaHMvq25nhAA1CA6Qqi9Sksd9wb7xz8cz2++WZo1S0pJsbYuDy62MfpcSVxHCAB8gi
CE2mndOsdZYd9847gq9KhR0l/+IoX653/SVd3wfHdqM93eNpkrSwOAj/jntwZwPhUV0osvOhZVT0tNm0rvvSt162ZlTZe
Enz4tNW3qWArr3t3qyi6oqhueb2+brNTmDWu4GgBAJYIQ/FpJZq/6/etttoyxVvcgtqVtqv6lqmr2VgsuWsOoXBqo
sqoq6ZLSQMmxESooLvO4T4iN0QBgDTZLo3Z4/33phnhhscISgqSnr7bccm6VoQgiQpJDhImX1bS
zqzEboSG6MBwDoEIfi3Y8ekBx5wH6yBGpc2cpN9dtlhUZNUHr4rfhUaerdN1tSBHZUU67pMlhQboa
kDO7IxGgAswNIY/FdurpSRIe3c6Qg9I0dKY8dK4eFWV3bJerdNVs/WSVwZAgB8AAQh1A6Lt6FBIPmeEFnNJ9lB1bJ6bS1Pz7Q1XeLJ1Q3lBEfnhr1Sb2ScLQnIH0SPqQ6lCcNYVOUD7h5Pb6zmiQOepvKqMHqKeV8ug5o2b3zH6bMn8nM1FnNKAMCREtGuHAgrFXsw4rj+bGEmxsxT0DCxzFvGTmMbPfXqhWYdIJS3QWqkqi+EwpRPDQ1uJnQY3cfwDtJk2DxYdyM0bbsQaAAsIyK+JOtMmpGDp3HAlKlvlHD73brr4hu2kXZ7NUfDq2lC7FwqB3fZzWqT9s+NA7gfd3HBcgDH0e2FT9X3U8B6jc2lIyz0aSx1dnHM5E9T0+ALMqDnkEtB4sxCdg0uI9ZNWzqXl9CyFUWbwFqV4wAFkZcgZ3FdC8pqBgzx6SH0BbbwEyAgzwp/zB1Y8xkGbUDRpaDnZ4GqzPoGzBUZg/kPqHeq7W6Q/z3Z/cp5hQZhpYkWGAjjxZm9iYBGHGvCWnxJRCu+9qImZDEyrEbWHAgkqfjDQQxZ3NtqQXUWlOjDRt3YsvXTjXG2T5nEQnDKByd9zaOIn3p0ah5sGV/ArTMWUDhgVHfC+3lGHhxYzTf0ZEBOOHsENyEvA8yBdNoWZAGsb0AqSgpBzCj4dpPVZtPSlR0V90UuVAUiqsHR9LHlDDJuCk1ZHaHA6u6tWeXiXdVvFm6H60Z1A0ip1cPY8Z2sYbtGzw5+eGoYmqMvcMrsxgqJvU1dOIVStd+hS/5e++p81Z1M2S/6BLxd0Pd9vFNzeW4cjIQQUQnlBuFrxVUzdHqxdW6pSBkIuZsdT8Ngu8zjMcBXXYkkFDJMjY6kkZ3EXOjQjwWd16DWAD9bDUNBESLfuyXBoy00KSz0Hj3ZK96TsWLbT0csJZb3KH6P3cuJXfOt5Bj83rIjVvBnphKAY+nmc5gNeyIsrMn7BrR6FTeX2VSqGZfkXGs6xZcEqY+IVVQ1M2ErAoKDchbyCdvqiEZULq3sDjCgU+80RNlq6fOEjMNwV3mP0TZt5g9PLwrIe6I+1Jkf9tflLrsmsknTp2yzKAS2rUx/Vdyg5TJAgnrEGgUxFDJt6rb3Ydb4mC0J2L3/iL2g/F3+s8FDkxETdVHbBm9WuYdLaYcmSL/K4Uqyzy1IbgGDeAAfgQQpWYswKdiVbGXD8N6L1DfoJ8bS1FQq5MRgmrPZl3FaqXJQTLU9CTKl1oPWPyl5q9E/TUndRHzPTIfA7q1v1gV4zZyQjRGa3rCXnNT4aJLAFEXwKJH/RqWZRKJZlbWCdSexKr0kkfOdmZ6SC79W3gSgmKRM9Oc5uuZfwcbrOoV5hTDzMpFh5VXLjNnx7yDpwsJXHfGmWcXxYtQnoT8gsEMDFRaqBNSNMS6BMyxO1AnU3MZrg6yZ3xJFGnMvRPwXlFEGSXkWZVVEAymoZ0GPEvp7JfAqjx

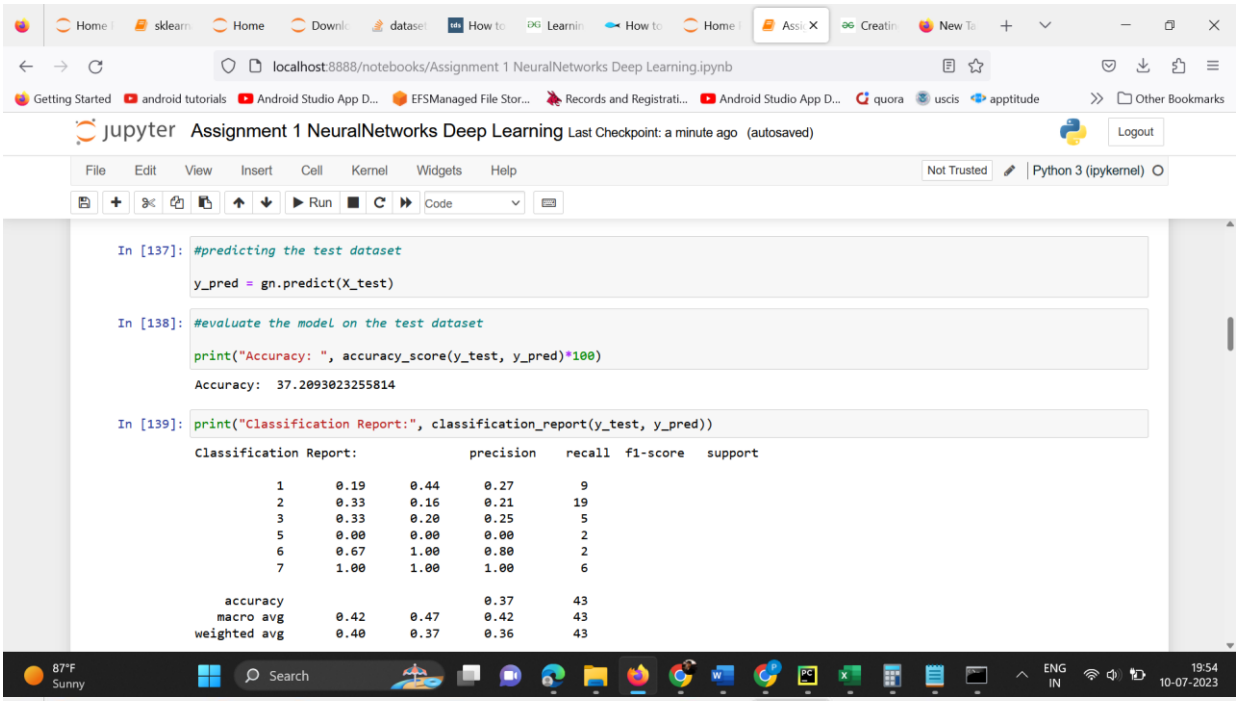
XF0rIAAPAGOkI4v3XrHKfFz5wpBQdLo0ZJn39OCAIA1Bl0hOCuosJxW4xRoxybo5s0cWyIvuU
WqysDAMCrCEJw9d//SnffLa1Y4Xjev780bZoUH29tXQAA1ACWxnDG++87bpa6YoXjqtB//7tj
kzQhCABQR9ERgnTsmPToo9KMGY7nnTs7NkS3aGFtXQAA1DA6QoEuN9cRfGbMkIKCpJEjHafIE
4IAAAGAjlCgstulv/1NevJJ6eRJqVEjx3WBevSwujIAAHyGIBSICgqkwYOlTz5xPO/XT3rrLa
lhQ0vLAgDA11gaCzSLFjk2RH/yiRQZ6Tgj7P33CUEAgIBERyhQlJVJTzwhvfaa43n79o4N0a1
bW1sXAAAWoiMUCPLypJ/85EwIGjFC+uILQhAAIODREarLjJGmTJH+/GdHRyghQXrnHal3b6sr
AwDALxCE6qrDh6V775UWLnQ8v/126e23pcEa7TpNftIgQB
ADAOegI1SUnT0rPPCO9+KLjeevW0nvvOUIRAABwQxCqK3bulDIyHFeKlqRhwxyBqF49a+sCAM
CPsTRW2xnjuBhix46OENSwofTRR45N0oQgAAAuiI5QbVZYKD3wgPSvfzme9+jhue1Go0bW1gU
AQC1BR6i2WrnScVHEf/1LCg2VJk50XC2aEAQAQJXREaptp2SxoyRxo1zLIu1aOHYEN2pk9WV
AQBQ6xCEapNvvpF+/3tp3tTrH8yFDpFdflerXt7QsAABqK5bGagNjpHfflW680RGC2uKefMcm
6QJQQAAXDI6Qv6uuFj64uBWbNkpo2tbYuAADqADpC/mztWkcXAM4cKSX
UKQhAAAF5CR8gfnT4tvfCC9OyzUkWFlJLiCEM33WR1ZQAA1CkEIX+zZ480cKC0erXj+cCB0ht
vSDEx1tYFAEAdxNKYP5k3z3Ftv6Tg6QlZbv17q0MERgoKDpVGjpFWrCEEAAPgAHSGr2O3SpEnSX/7i2BzdpInjt
PhbbbBW6MgAAAgZByAr79kl33y0tX+54/pvfSNOnS/Hx1tYFAECAqfGlsfHjxysoKEgjRoxwjp
WVlWn48OFq2LCh6tevr7vuuksHDx50+bnvvvtOffr0UvbsOffr0Ub169ZSQkKDHH39cp0+fdpnz2WefqWP
HjrLZbLr22ms1c+ZMt/d/dPXVysiIkJdu3bVusrbU1gpP1/67DMpKkr6+9+lf/6TEAQA
gAVqNAitX79e06dP14dP1w033OAy/sgjj2jEtt93+vOO+90Hq+oqFCfPn108uRJr
V27Vu+8845pyp0a0NHO+fk5+erT58+ueuqOrrM3dT2/DLoL1cdmEj4x7R06VLnnnH/+85969NFZ
mZqY0bN6p9+ZKT0/XoUOHdbWWAwBAoDI1pLS01LLRo0cJqXk
Z2267zTz88MPGG0KiopmwMxlnK+/x67e//SvVpycVysP/VKpVsTPNFZ2V/hEHhx4erT58+SktLcxnfsGGDTp065DTp/CfVU
xstWvXTomJic456enpkip0bEt25xzzn3t9PR052ucPHlSGzStjLTHBystLQ055xzlZeXq6S
kxOUBAADqrhrZLD137lxt3LhLE6evXfN0+qeeodKHjDlQtQXAANRqXu8I7d27Vw8Vw8//
lJToxIkTOnLkiCoqKjzO2bdSP6ZR/d2Tb2Bi6ygQh4f4v9kEn6nQ6E
UmJurkyZMqKipy+bmDwNsQzg/QoSlKqlt1cQyJkpSUlOR2Flnl84vNiYmJUnJUTRkpK644GqFhIR4nFP5Guey2Wy
KiYlxeQAAgLrL60Gp0GrrqZz8x5M0uqDaHb1NRb/AveVqepS1Lx8Gs0oKgXeZMmvSR7Eh4JxTeP5Gu
V2Wy
SVJqaqq2bt3qcnZXla1Lp1a+ecs1fFG0M77xj7fq0+xuzvrMT47Zw4ADjxzfVbU1rNuxAUS6lfhp
z/oo037lP3ND6qwG0vrAQAgUHl9j1B0dLQuvLCC9Jf+d+FRXVVrm19R0nxH0MGxb3/ffTcW1M27qd
SG96fz0Z+f7ZNvuKcGdHJTcHaWGPrGqqwSa6NLxpqpI2VGt4TVW+r/qjhKJAPd5b3sgvkD4ldWDAM/yzO
n8w7UPozm1+vQKu2Yo7KqNDj2Vo0kUq86IbKqWkwsRAq7JyeQ6SZY7l9kq/W86lqpX0rDgTvTe86lpbW
kXMAqKrZjr2A0//Gt4p8G0ye38n64k8Re7LJV7Fo0r3LDa5VpPPtMrbmRJkTBTcmjRCbTqYkRlm3tpxS
6eXOhjYuf9tPefpzhQRXH8fPlXtq5QqW5pa5M1H6NFT9iRBQYsLU0Er6Jvd1zKZLvI7oq2Fz4Se1d1R
O2MtbSVa8s5Y1/xXyZ5B9JD6ycWq5uiEhA/jBtfA5m6kxsYm8jgkhDuD1z+/nOm7OyOakD6SS+kGvzB
DPiLGKdK5XdtvF7zQs/ExPeitz0wfHXTaTUMbzI0J1p58hLUTxgm6pJyVmR3aSIOuAgeD1dngXqH+YGSe
hO0A0Az6MIWi5T9mS37gVKT67xx0cbvgHNFaGnxntdudGtTwjlO5LwOHxLxaJu7GLjmlxfsKBkpMHBD
bsFp9gJyqPyJcl1GM3P8pbqAYcVe9P2GMIjOEkIl3C4DMg/SPPjQKPaFBrCA9Njc5XlhDvVVQE2BvTn
6e4vnIx8RVAUN3SbW9ai+2cYPgGGEtAJ6T8XYuWEW+bOZb8KEM2W3GzFQGvDpTFLNB5IGSRdrfjhb8ND
GQgeKb/Axxjb5oPOkGPwwEJRtS2S2oYXVIcEVm1MQpoEDCe51fcxNupymlYJTVTBzs0o2y+hn0uSqhh
VF0S8z3/T+vH3YW8ipz3LpK4aADQHFArX4cbWjJm3MjP1M2DtFbjEq9S5ifVXx2L5nxnKs05mXhaNQJm
t9adM+Wm87I/dqJ14Fmk94dEWabXJbrtkPKVq4vUHELjVXPlOqmB3lIn8JQHUh/4Xt08uFwSKVmqeUp
5tLKsz2L1jmEeHb8/v9bTJ7dWL3T85YJj0j3QMMNqgTMStpnUXk/jwmO3e2/uE36jZ7NQY/p5o5wr2
lTTkM1sQcLLr4hT9r5e46OxOlB2cIhBDEQSHhtLoXBVrvXo3o4UZHafHu5j/v+QxV2kaXxZQiAhZJ2
AMfjI8fdcmpbXDDiZtoSNPgx3Umk4e5z7JmDBFGUQFAYY8VqFA8jV/2o1AqBmeQN+3m2lHkFqQGmHH
NWh+8nKEiIHcdOp6kORvdmbBpI7K5P0RXHjL1q7oYAv2TJHqp0yDmUkAQIThz7LkgAIAAAPI0wbK5B
8c9iz3OWBK0jpQMjSFbj/CJ/81uVcBYr9mAmvx0m4WO1iMhG9sH4gz7qBJ6F3iFHtJmSOxs7a4vLBa
rF9bQmj7Fd4X5Xz8IhMlu09F0gF/cN3yFdLAkAGvC3I9h+TCRIb9l4jKuTYFxYFU7mS6BZHL2AuB5mW
jHJo0cVTP4FkjMuIzWnW9ugv+NJMNV9kxzRPlgJdd5AMP0JB7KuH16DGhVzyW3dEk/sSEsdjQMAAH3
g8HLh47UMkvYTNQCkGgaqkJQ5i2W4EAvfJ9sG8uKkg3Il2vvgyyBcFmBmWTqU5VTyVSqoz4y71EJmKJ

```
4MGDrS4voD355JMqKSlRq1atFBISooqKCj3//PMaMGCA1aXhRwUFBZKkxMREl/HExETnMX9GE
Krjhg8frry8PK1evdrqUgLe3r179fDDDysrK0sRERFWl4Oz2O12de7cWS+88IIkqUOHDsrLy9
O0adMIQhabN2+eZs+erTlz5qhNmzbatGmTRowYoUaNGvG7gVewNFaHPfjgg1q4cKFWrFihq66
6yupyAt6GDRt06NAhdezYUaGhoQoNDdXKlSs1efJkhYaGqqKiwuoSA1ZycrJat27tMnb99dfr
u+++s6giVHr88cf15JNP6ne/+53atWunQ6ne/+53atWunQYMG6ZFHHHtG4ceOsLg0/SkpKKkiQdPHjQZfzgwYPOY
/6MIFQHGWP04IMP6oMPPPtDy5cuVkpJidUmQ1KNHD23dulWbNm1yPjp37qwBAwZo06ZNCgkJsb
rEgNWtWze3S0x8/fXXatasmUUVodLx48OzVHz6VRUSEiK73W5RThXSkqKkpKStGzZMudYSUm
JcnJylJqaamFlVcPSWB00fPhwzZkzRx999JGio6Oda7SxsbGKjIy0uLrAFR0d7bZPPKyoqSg0b
NmT/lsUeeeeQR3XzzzXrhhRfUv39/rVu3TjNmzNCMGTOsLi3g9e3bV88//7aNm2qNm3aKDc3V
y+//LLuvfdeq0sLKEePHtXu3budz/ft/Pz87Vp0yY1aNATZs21YgRI/TXv/5VLVq0UEpKikaNGq
VGjRo5zyyzzawZ1jiSPj7ffftvq0nCO2267zTz88MNWlwFjzIIIFC0zbtm2NzWYzrVq1MjNmzLC
6JBhjSkpKzMMPP2yaNm1qIiIizDXXGOeeeYZU15ebnVpAWXFihDx5sjDHGbrebUaNG
mcTERGOz2UyPHj3Mzp07rS26iriOEAAACFjsEQIAAAGLIAQAAAIWQQgAAAQsghAAAAhYBCEAA
BCwCEIAACBgEYQAAEDAIggBAICARRACAAABiyAEAAACFkEIAAAErP8PKV8riGlhMS4AAAAASU
VORK5CYII=",
      "text/plain": [
       "<Figure size 640x480 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "# 3(a) Import the given “Salary_Data.csv”\\n\",\n",
    "import numpy as np\n",
    "dst_Sal = pd.read_csv(\"Salary_Data.csv\")\n",
    "dst_Sal.info()\n",
    "dst_Sal.head()\n",
    "\n",
    "A = dst_Sal.iloc[:, :-1].values   #excluding last column i.e., years
of experience column\\n\",\n",
    "B = dst_Sal.iloc[:, 1].values     #only salary column\"\n",
    "\n",
    "# (b) Split the data in train_test partitions, such that 1/3 of the
data is reserved as test subset.\\n\",\n",
    "from sklearn.model_selection import train_test_split\n",
    "A_train, A_test, B_train, B_test = train_test_split(A, B,
test_size=1/3, random_state=0)   \n",
    "\n",
    "#(c) Train and predict the model.\\n\",\n",
    "from sklearn.linear_model import LinearRegression\n",
    "reg = LinearRegression()\n",
    "reg.fit(A_train, B_train)\n",
    "B_Pred = reg.predict(A_test)\n",
    "B_Pred\n",
    "    \n",
    "\n",
    "# (d) Calculate the mean_squared error\\n\",\n",
    "S_error = (B_Pred - B_test) ** 2\n",
    "Sum_Serror = np.sum(S_error)\n",
    "mean_squared_error = Sum_Serror / B_test.size\n",
    "mean_squared_error\n",
    "\n",
    "#(e) Visualize both train and test data using scatter
plot.\\n\",\n",
    "import matplotlib.pyplot as plt\n",
    "# Training Data set\n",
```

```
    "plt.scatter(A_train, B_train)\n",
    "plt.plot(A_train, reg.predict(A_train), color='red')\n",
    "plt.title('Training Set')\n",
    "plt.show()\n",
    "# Testing Data set\\n\",\n",
    "plt.scatter(A_test, B_test)\n",
    "plt.plot(A_test, reg.predict(A_test), color='red')\n",
    "plt.title('Testing Set')\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "ba91b995",
   "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": "3.11.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```


localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb
jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [131]:
#1.Implement Naïve Bayes method using scikit-learn library
#Use dataset available with name glass
#Use train_test_split to create training and testing part
#Evaluate the model on test part using score and classification_report(y_true, y_pred)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
import warnings
from sklearn import metrics
In [132]:
#importing the given dataset glass.csv
dst_Data = pd.read_csv("glass.csv")
dst_Data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 RI 214 non-null float64
1 Na 214 non-null float64
2 Mg 214 non-null float64
87°F Sunny
Search
ENG IN
19:53 10-07-2023


localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb
jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
In [133]:
#splitting the dataset which is excluding last columns
X = dst_Data.iloc[:, :-1]
y = dst_Data.iloc[:, -1]
In [134]:
#splitting the dataset into train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
In [135]:
#creating a Gaussian Naive Bayes model
gn = GaussianNB()
In [136]:
#fitting train data
gn.fit(X_train, y_train)
Out[136]:
GaussianNB
GaussianNB()
87°F Sunny
Search
ENG IN
19:54 10-07-2023



Home | sklearn | Home | Downlc | dataset | How to | Learnin | How to | Home | Assi | Creatin | New Ta

localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb

Getting Started | android tutorials | Android Studio App D... | EFSManaged File Stor... | Records and Registrati... | Android Studio App D... | quora | uscis | apptitude | Other Bookmarks

jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [137]:
```
#predicting the test dataset

y_pred = gn.predict(X_test)
```

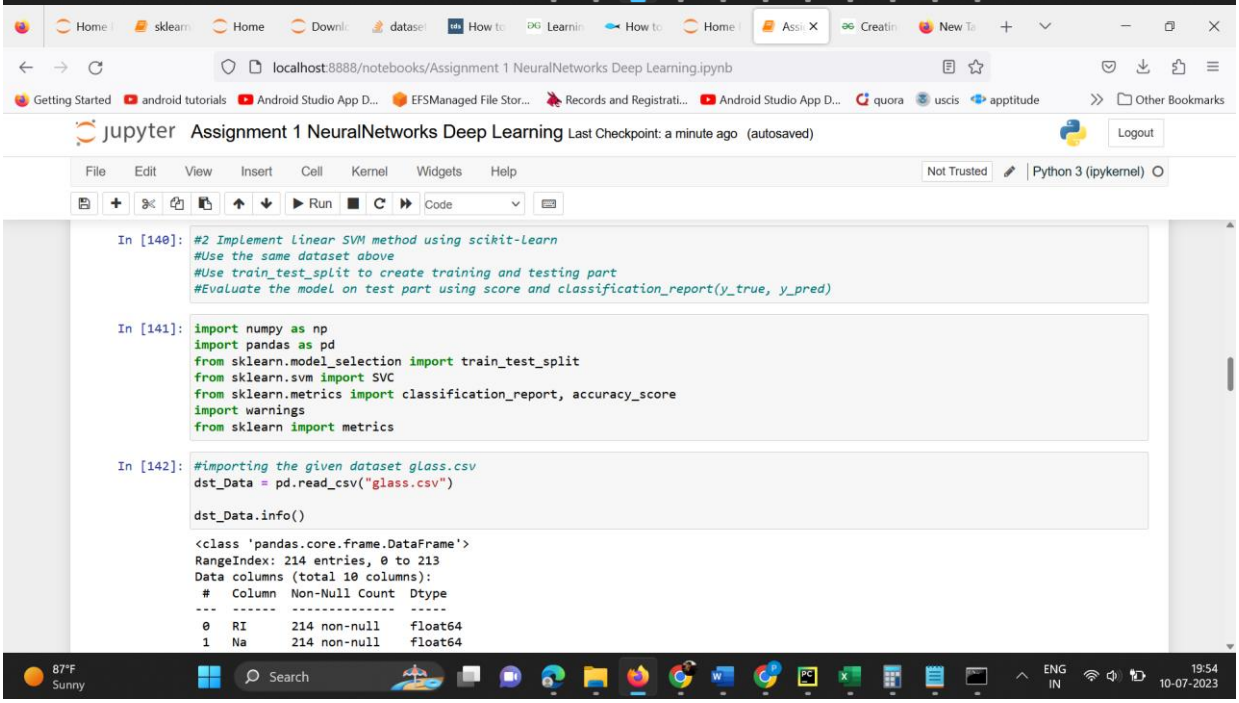
In [138]:
```
#evaluate the model on the test dataset

print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
```

```
Accuracy:  37.2093023255814
```

In [139]:
```
print("Classification Report:", classification_report(y_test, y_pred))
```

```
Classification Report:               precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6

    accuracy                           0.37        43
   macro avg       0.42      0.47      0.42        43
weighted avg       0.40      0.37      0.36        43
```

87°F Sunny | Search | ENG IN | 19:54 10-07-2023



Home | sklearn | Home | Downlc | dataset | How to | Learnin | How to | Home | Assi | Creatin | New Ta

localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb

Getting Started | android tutorials | Android Studio App D... | EFSManaged File Stor... | Records and Registrati... | Android Studio App D... | quora | uscis | apptitude | Other Bookmarks

jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

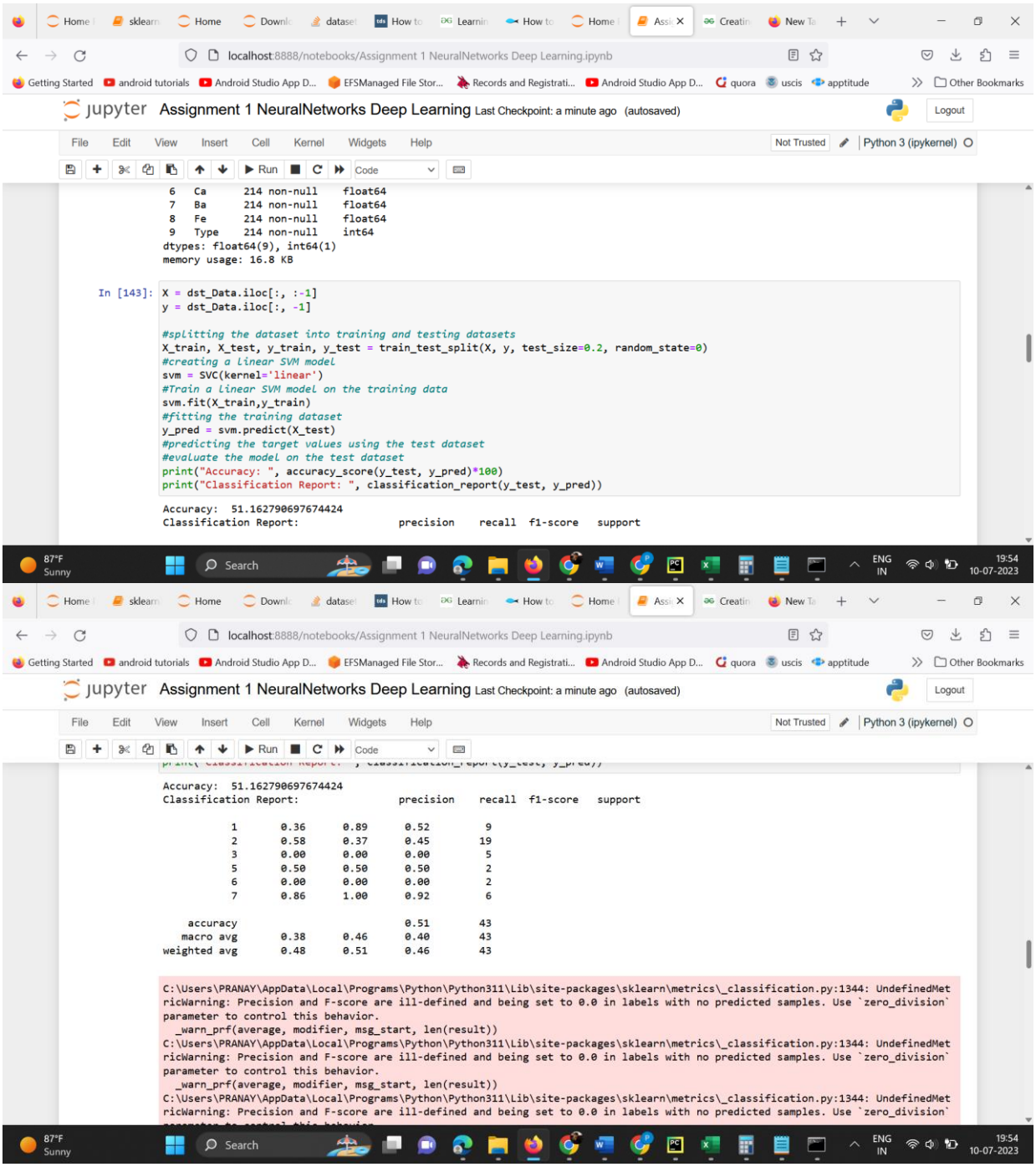
In [140]:
```
#2 Implement linear SVM method using scikit-Learn
#Use the same dataset above
#Use train_test_split to create training and testing part
#Evaluate the model on test part using score and classification_report(y_true, y_pred)
```

In [141]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
import warnings
from sklearn import metrics
```

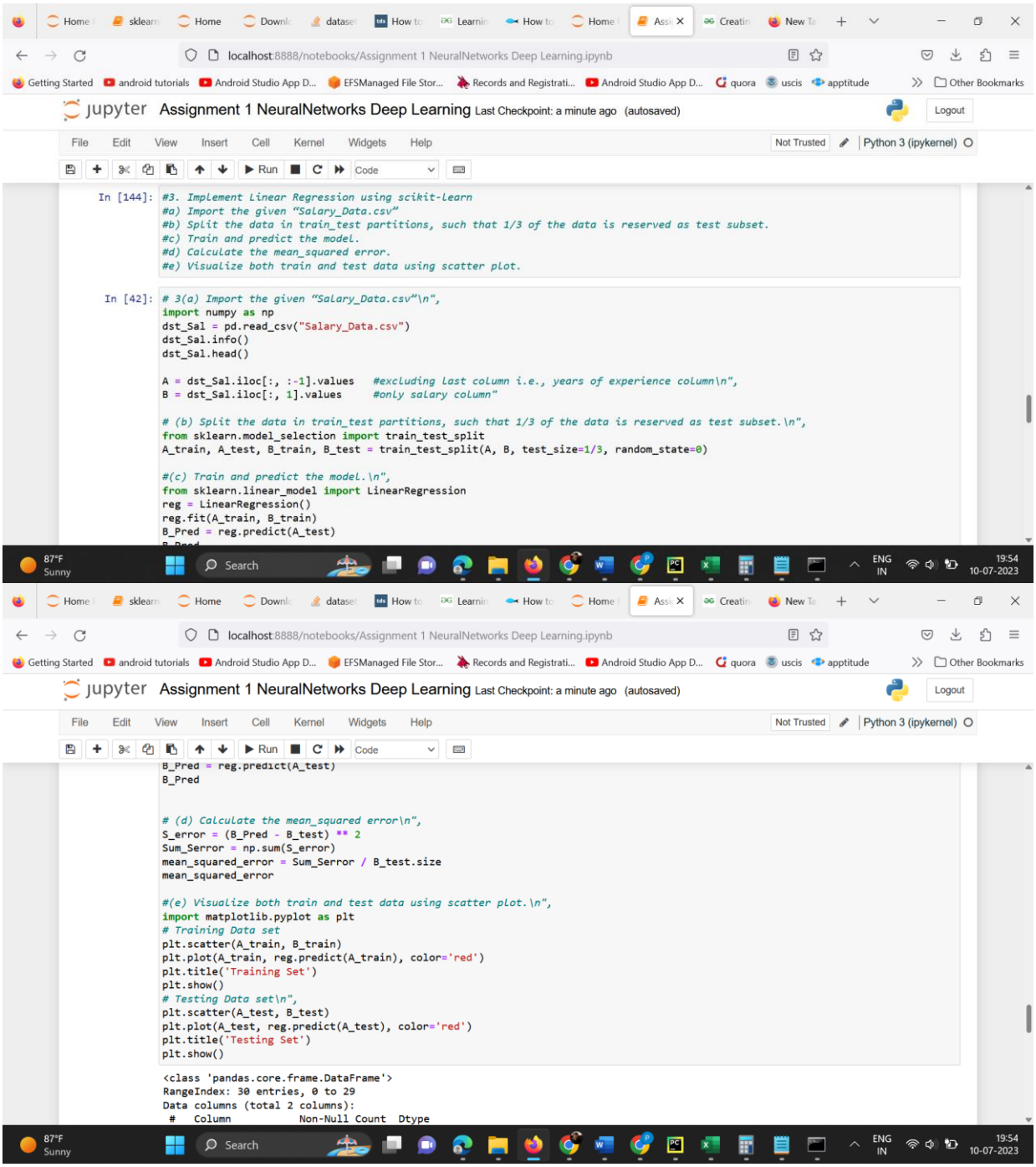
In [142]:
```
#importing the given dataset glass.csv
dst_Data = pd.read_csv("glass.csv")

dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
```

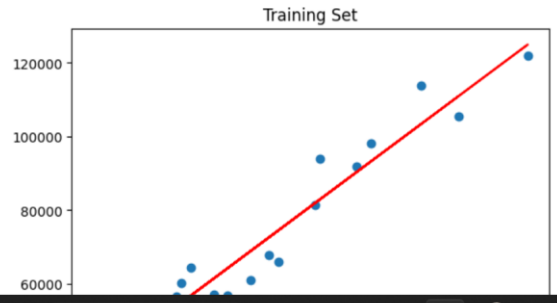
87°F Sunny | Search | ENG IN | 19:54 10-07-2023


jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
6 Ca 214 non-null float64
7 Ba 214 non-null float64
8 Fe 214 non-null float64
9 Type 214 non-null int64
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
In [143]:
X = dst_Data.iloc[:, :-1]
y = dst_Data.iloc[:, -1]
#splitting the dataset into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
#creating a linear SVM model
svm = SVC(kernel='linear')
#Train a linear SVM model on the training data
svm.fit(X_train,y_train)
#fitting the training dataset
y_pred = svm.predict(X_test)
#predicting the target values using the test dataset
#evaluate the model on the test dataset
print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
print("Classification Report: ", classification_report(y_test, y_pred))
Accuracy: 51.162790697674424
Classification Report: precision recall f1-score support
87°F Sunny
Search
ENG IN
19:54 10-07-2023

jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Accuracy: 51.162790697674424
Classification Report: precision recall f1-score support
1 0.36 0.89 0.52 9
2 0.58 0.37 0.45 19
3 0.00 0.00 0.00 5
5 0.50 0.50 0.50 2
6 0.00 0.00 0.00 2
7 0.86 1.00 0.92 6
accuracy 0.51 43
macro avg 0.38 0.46 0.40 43
weighted avg 0.48 0.51 0.46 43
C:\Users\PRANAY\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\PRANAY\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\PRANAY\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division`
87°F Sunny
Search
ENG IN
19:54 10-07-2023


localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb
jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [144]:
#3. Implement Linear Regression using scikit-Learn
#a) Import the given "Salary_Data.csv"
#b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
#c) Train and predict the model.
#d) Calculate the mean_squared error.
#e) Visualize both train and test data using scatter plot.
In [42]:
# 3(a) Import the given "Salary_Data.csv"\n",
import numpy as np
dst_Sal = pd.read_csv("Salary_Data.csv")
dst_Sal.info()
dst_Sal.head()
A = dst_Sal.iloc[:, :-1].values #excluding last column i.e., years of experience column\n",
B = dst_Sal.iloc[:, 1].values #only salary column"
# (b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.\n",
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=1/3, random_state=0)
#(c) Train and predict the model.\n",
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(A_train, B_train)
B_Pred = reg.predict(A_test)
87°F Sunny
Search
ENG IN
19:54 10-07-2023
B_Pred = reg.predict(A_test)
B_Pred
# (d) Calculate the mean_squared error\n",
S_error = (B_Pred - B_test) ** 2
Sum_Serror = np.sum(S_error)
mean_squared_error = Sum_Serror / B_test.size
mean_squared_error
#(e) Visualize both train and test data using scatter plot.\n",
import matplotlib.pyplot as plt
# Training Data set
plt.scatter(A_train, B_train)
plt.plot(A_train, reg.predict(A_train), color='red')
plt.title('Training Set')
plt.show()
# Testing Data set\n",
plt.scatter(A_test, B_test)
plt.plot(A_test, reg.predict(A_test), color='red')
plt.title('Testing Set')
plt.show()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
Column Non-Null Count Dtype

```
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```



localhost:8888/notebooks/Assignment 1 NeuralNetworks Deep Learning.ipynb

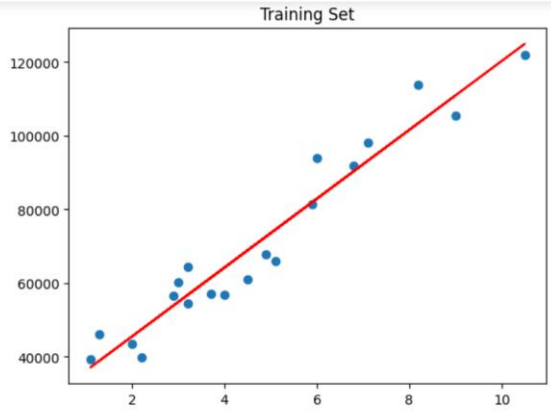
jupyter Assignment 1 NeuralNetworks Deep Learning Last Checkpoint: 2 minutes ago (autosaved)

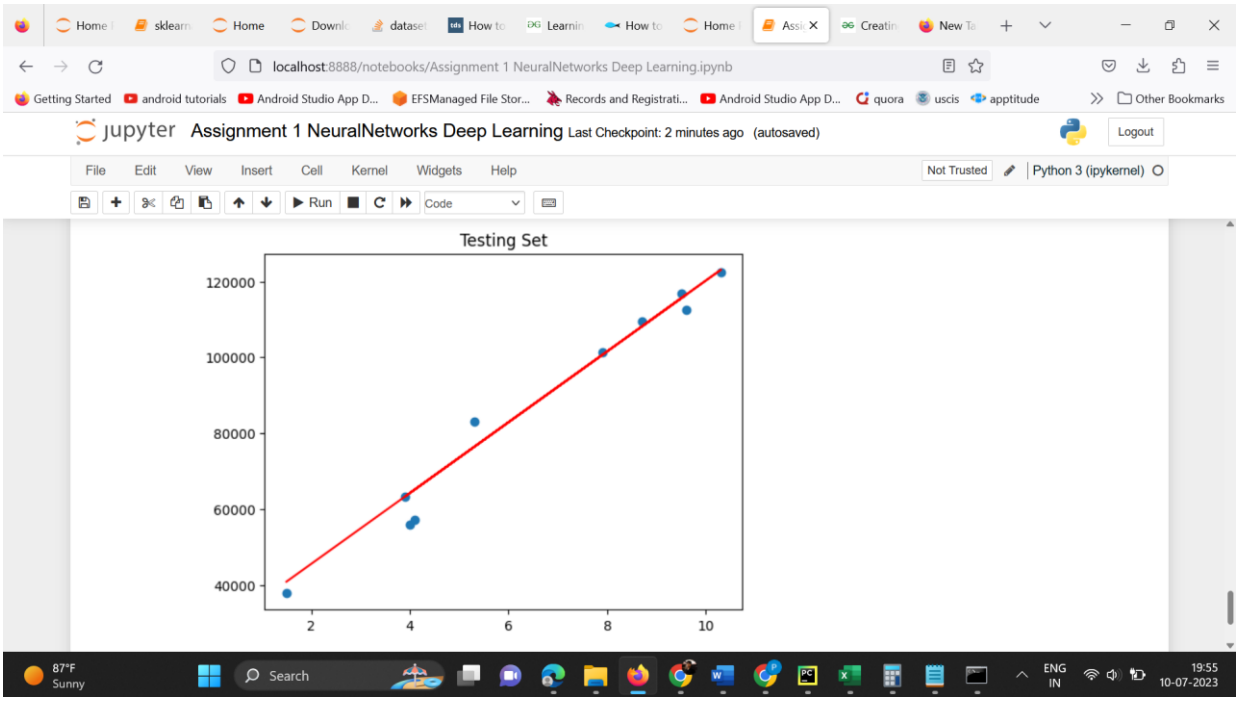
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code





In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
import warnings
from sklearn import metrics
```

In [3]:
```
dst_Data = pd.read_csv("glass.csv")
dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
```

jupyter Assignment 1 NeuralNetworks DL Last Checkpoint: 13 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [4]:
```
X = dst_Data.iloc[:, :-1]
y = dst_Data.iloc[:, -1]
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [6]:
```
gn = GaussianNB()
```

In [7]:
```
gn.fit(X_train, y_train)
```

jupyter Assignment 1 NeuralNetworks DL Last Checkpoint: 13 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

Out[7]:
```
GaussianNB
GaussianNB()
```

In [8]:
```
y_pred = gn.predict(X_test)
```

In [11]:
```
print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
```

```
Accuracy:  37.2093023255814
```

In [15]:
```
print("Classification Report:", classification_report(y_test, y_pred))
```

```
Classification Report:               precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6
```

jupyter Assignment 1 NeuralNetworks DL Last Checkpoint: 13 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
                accuracy                           0.37        43
               macro avg       0.42      0.47      0.42        43
            weighted avg       0.40      0.37      0.36        43
```

In [16]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
```

In [17]:
```
dst_Data = pd.read_csv("glass.csv")
dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
```

jupyter Assignment 1 NeuralNetworks DL Last Checkpoint: 13 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
y_pred = svm.predict(X_test)
print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  51.162790697674424
Classification Report:                precision    recall  f1-score   support

           1       0.36      0.89      0.52         9
           2       0.58      0.37      0.45        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.51        43
   macro avg       0.38      0.46      0.40        43
weighted avg       0.48      0.51      0.46        43
```

```
: import pandas as pd
  from sklearn.model_selection import train_test_split
  from sklearn.linear_model import LinearRegression
  from sklearn.metrics import mean_squared_error
  import matplotlib.pyplot as plt

  # Step a: Import the dataset
  data = pd.read_csv('Salary_Data.csv')
  X = data.iloc[:, :-1].values
  y = data.iloc[:, -1].values

  # Step b: Split the data into train and test subsets
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=42)

  # Step c: Train and predict the model
  regressor = LinearRegression()
  regressor.fit(X_train, y_train)
  y_pred_train = regressor.predict(X_train)
  y_pred_test = regressor.predict(X_test)

  # Step d: Calculate the mean squared error
```