```json
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Module 3: Data Exploration\n",
    "\n",
    "The following tutorial contains examples of Python code for data exploration. You should refer to the \"Data Exploration\" chapter of the \"Introduction to Data Mining\" book (available at https://www-users.cs.umn.edu/~kumar001/dmbook/index.php) to understand some of the concepts introduced in this tutorial notebook. The notebook can be downloaded from http://www.cse.msu.edu/~ptan/dmbook/tutorials/tutorial3/tutorial3.ipynb.\n",
    "\n",
    "Data exploration refers to the preliminary investigation of data in order\n",
    "to better understand its specific characteristics. There are two key motivations for data exploration:\n",
    "1. To help users select the appropriate preprocessing and data analysis technique used.\n",
    "2. To make use of humansâ€™ abilities to recognize patterns in the data.\n",
    "\n",
    "Read the step-by-step instructions below carefully. To execute the code, click on the cell and press the SHIFT-ENTER keys simultaneously."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## 3.1. Summary Statistics\n",
    "\n",
    "Summary statistics are quantities, such as the mean and standard deviation, that capture various characteristics of a potentially large set of values with a single number or a small set of numbers. In this tutorial, we will use the Iris sample data, which contains information on 150 Iris flowers, 50 each from one of three Iris species: Setosa, Versicolour, and Virginica. Each flower is characterized by five attributes:\n",
    "\n",
    "- sepal length in centimeters\n",
    "\n",
    "- sepal width in centimeters\n",
    "\n",
    "- petal length in centimeters\n",
    "\n",
    "- petal width in centimeters\n",
    "\n",
    "- class (Setosa, Versicolour, Virginica) \n",
    "\n",
    "In this tutorial, you will learn how to:\n",
    "\n",
    "- Load a CSV data file into a Pandas DataFrame object.\n",
    "\n",
    "- Compute various summary statistics from the DataFrame.\n",
    "\n",
    "To execute the sample program shown here, make sure you have installed the Pandas library (see Module 2)."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**1.** First, you need to download the <a href=\"http://archive.ics.uci.edu/ml/datasets/Iris\">Iris dataset</a> from the UCI machine learning repository.\n",
    "\n",
    "**<font color='red'>Code:</font>** The following code uses Pandas to read the CSV file and store them in a DataFrame object named data. Next, it will display the first five rows of the data frame."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import pandas as pd\n",
    "\n",
    "data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',header=None)\n",
    "data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'class']\n",
    "\n",
    "data.head()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**2.** For each quantitative attribute, calculate its average, standard deviation, minimum, and maximum values.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from pandas.api.types import is_numeric_dtype\n",
    "\n",
    "for col in data.columns:\n",
    "    if is_numeric_dtype(data[col]):\n",
    "        print('%s:' % (col))\n",
    "        print('\\t Mean = %.2f' % data[col].mean())\n",
    "        print('\\t Standard deviation = %.2f' % data[col].std())\n",
    "        print('\\t Minimum = %.2f' % data[col].min())\n",
    "        print('\\t Maximum = %.2f' % data[col].max())"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**3.** For the qualitative attribute (class), count the frequency for each of its distinct values.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "data['class'].value_counts()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**4.** It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values. \n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "data.describe(include='all')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Note that count refers to the number of non-missing values for each attribute."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**5.** For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "print('Covariance:')\n",
    "data.cov()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "print('Correlation:')\n",
    "data.corr()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## 3.2. Data Visualization\n",
    "\n",
    "Data visualization is the display of information in a graphic or tabular format. Successful visualization requires that the data (information) be converted into a visual format so that the characteristics of the data and the relationships\n",
    "among data items or attributes can be analyzed or reported.\n",
    "\n",
    "In this tutorial, you will learn how to display the Iris data created in Section 3.1. To execute the sample program shown here, make sure you have installed the matplotlib library package (see Module 0 on how to install Python packages)."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**1.** First, we will display the histogram for the sepal length attribute by discretizing it into 8 separate bins and counting the frequency for each bin.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "%matplotlib inline\n",
    "\n",
    "data['sepal length'].hist(bins=8)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**2.** A boxplot can also be used to show the distribution of values for each attribute.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "data.boxplot()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**3.** For each pair of attributes, we can use a scatter plot to visualize their joint distribution.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import matplotlib.pyplot as plt\n",
    "\n",
    "fig, axes = plt.subplots(3, 2, figsize=(12,12))\n",
    "index = 0\n",
    "for i in range(3):\n",
    "    for j in range(i+1,4):\n",
    "        ax1 = int(index/2)\n",
    "        ax2 = index % 2\n",
    "        axes[ax1][ax2].scatter(data[data.columns[i]], data[data.columns[j]], color='red')\n",
    "        axes[ax1][ax2].set_xlabel(data.columns[i])\n",
    "        axes[ax1][ax2].set_ylabel(data.columns[j])\n",
    "        index = index + 1"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**4.** Parallel coordinates can be used to display all the data points simultaneously. Parallel coordinates have one coordinate axis for each attribute, but the different axes are parallel to one other instead of perpendicular, as is traditional. Furthermore, an object is represented as a line instead of as a point. In the example below, the distribution of values for each class can be identified in a separate color.\n",
    "\n",
    "**<font color=\"red\">Code:</font>**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from pandas.plotting import parallel_coordinates\n",
    "%matplotlib inline\n",
    "\n",
    "parallel_coordinates(data, 'class')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## 3.3. Summary\n",
    "\n",
    "This tutorial presents several examples for data exploration and visualization using the Pandas and matplotlib library packages available in Python. \n",
    "\n",
    "**<font color='blue'>References:</font>**\n",
    "\n",
    "1. Documentation on Pandas. https://pandas.pydata.org/\n",
    "2. Documentation on matplotlib. https://matplotlib.org/\n",
    "3. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. "
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "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": "3.6.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```