ISAT 341: Machine Learning and Data Science

Project: Machine Learning Confidential Sensor Data

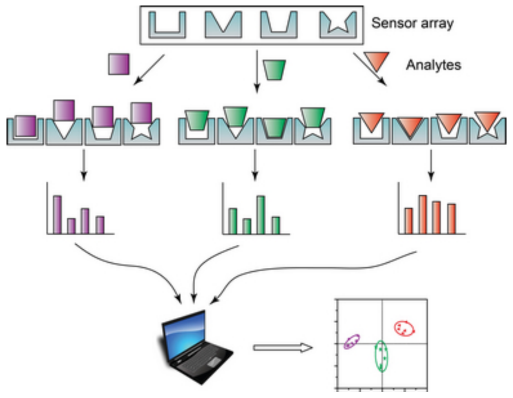
Working with *real-world* datasets

Objectives

To demonstrate the ability to complete an end-to-end data science / machine learning project using real-world data by following and implementing the main machine learning checklist steps that lead to a solution, namely:

- Frame the problem and look at the big picture.
- Get the data.
- Explore the data to gain insights.
- Prepare the data to expose the underlying data patterns to Machine Learning algorithms.
- Explore many different models and short-list the best ones.
- Fine-tune your models and combine them into a great solution.
- Present your solution.
- Launch, monitor, and maintain your system

Frame the Problem



Sensor Data

The data source as well as the exact nature of the data is confidential. Each data instance contains 12 real-valued input attributes. Each input attribute represents a sensor designed to detect the presence of one of two groups of substances. As an alternative, the sensor readings may represent a 'false alarm'.

- Substance 1 is represented by the value 'one' in the class attribute column.
- Substance 2 is represented by the value 'two' in the class attribute column.
- A false alarm is represented by the value 'three' in the class attribute column.

The problem is framed as a **supervised learning** problem: Predict the class of a substance from sensor data using the given measurements in the dataset.

Project Analysis Starts Here!

DO NOT remove the TO DO labels and your Notebook MUST have the same numbering indicated here

In [1]:
```
# import packages
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
np.set_printoptions(precision=3, suppress=True)
```

Exploratory Data Analysis

1) TO DO: Use Pandas to load your data into a dataframe

Display the first ten rows after you load the data.

In [ ]:

2) TO DO: Use the dataframe describe method *dataframe.describe()* to display some summary statistics

In [ ]:

Comments on above summary of data

Examine the data and note that when a sensor temporarily goes offline, it records a BAD data value of -9999. This can be seen in the minimum recorded values for each sensor. Before we continue our preliminary data analysis, this bad data must be removed.

You should also take note of the **count**, **mean**, **standard deviation** and the **maximum** data values for each sensor. The bad data value recordings when the senors go *offline* have a large impact on these simple descriptive statistical summaries. Also, the wide range of the data values will require some sort of scaling of the data before training a machine learing model.

Finally, the class labels are not numeric but are strings ('one', 'two' and 'three') that we call categorical class labels. We will need to convert them to integers before we feed them to a classifier.

Data Cleansing

3) TO DO: Display the shape of your dataframe data in the cell below

In [ ]:

```

```

4) TO DO: Use Pandas dataframe to find bad or missing data.

The bad data can be located and replaced using pandas and numpy. In the cell below, use code to find and replace the bad data (-9999 values) with numpy nan values (nan means *not a number*). Then use the dataframe to drop the rows with the nan signatures. This may significantly reduce your dataset size so you should display the shape of the data to see its new size. Also, again, display the first ten rows of the data with the dataframe.

- **Enter your code in the cell below to implement the drop, replace and display**.
- **Enter your code in the cell immediately following the output of the dataframe display to show the new shape of data in the dataframe.**

In [ ]:

```

```

In [2]:

```
# shape of data in dataframe after cleansing
```

5) TO DO: Use pandas correlation method to find the two features (inputs) with the highest correlation

Enter your answers in the cell immediately following the output correlation table

```
In [ ]:
```

Your Answer:

Data Visualization

6) TO DO: Plot bar charts using pandas dataframe (plot the mean value of the sensors)

```
In [ ]:
```

Data Preprocessing

7) TO DO: Create Feature Matrix and Target Vector

"Features" are also known as predictors, inputs, or attributes. The **"response"** is also known as the target, label, or output.

complete in cell below

```
In [ ]:
```

8) TO DO: Convert the features dataframe to a numpy array

complete in cell below

```
In [ ]:
```

9) TO DO: Label Encoding

Transform the categorical labels into integers using the scikit-learn label encoder

Enter your code in the cell below

```
In [ ]:
```

10) TO DO: Split the data into Training and Testing Sets

Scikit learn contains a function called the **train_test_split** function that will randomly shuffle the dataset and then splits it into two datasets: a **training set** used to build the model and a **test set** to assess and evaluate how well the model works on unseen data (also called out-of=sample data).

Use a 80% / 20% train/test split for this project.

NOTE: You **must** use the encoded class labels in this part

Enter your code in the cell below

```
In [ ]:
```

11) TO DO:Look at the *shape* of the data (rows and columns) *after* splitting it into training and testing sets

Enter your code in the next TWO cells below

```
In [ ]:
```

```
In [ ]:
```

Scale the Data

IMPORTANT: Standardizing the features:

Standardization of datasets (feature scaling) is a common requirement for many machine learning and optimization algorithms implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data, i.e., Gaussian with zero mean and unit variance.

12) TO DO: Let's use the StandarScaler from Scikit-learn to transform (*scale*) our feature

Enter your code in the cell below

```
In [ ]:
```

A comment on what the above code does

- Using the preceding code, we loaded the StandardScaler class from the preprocessing module and initialized a new StandardScaler object that we assigned to the variable sc.
- Using the fit method, StandardScaler estimated the parameters μ (sample mean) and σ (standard deviation) for each feature dimension from the training data.
- By calling the transform method, we then standardized the training data using those estimated parameters μ and σ.
- Note that we used the same scaling parameters to standardize the test set so that both the values in the training and test dataset are comparable to each other.

IMPORTANT: So from this point forward you *must* use the *scaled* training and testing data

Model Building

Train multiple Machine Learning models during same session. You will use the three algorithms from previous labs.

1. K-Nearest Neighbor (with K=10, K=50, K=200)
2. Logistic Regression
3. Linear Support Vector Classifier

During the evaluation phase you must predict class member propbabilities and comment on what they mean.This means that scikit-learn's linear support vector classifier cannot be used for those prodictions since this model does not have a method to predict class membership probabilities, but like the other models, it does have the same *predict* method.

- You MUST properly use model evaluation metrics (accuracy, confusion matrix, etc.)
- Two of your models (KNN and Logistic Regression MUST predict class membership probabilities and associated class label names

Build a KNN Classification Model for K = 10, 50 and 200

13) TO DO: In the sections below you should build and train the actual machine learning model.

In the cell below: enter the code to import, instantiate, fit, predict and test the model's performance (accuracy) for the *K-Nearest Neighbor* Model in SciKit-Learn for K= 10, 50 and 200. *This should be completed in ONE cell using a loop, etc.*

```
In [ ]:
```

Predicting class-membership probabilites

Scikit-Learn has a method that allows prediction of *class member probabilities*.

- The probability that training examples belong to a certain class can be computed using the **predict_proba** method. For example, we can predict the probabilities of the first three samples in the test set as follows (NOTE: X_test_std[:3. :] means get the first 3 rows and the associated columns from the test dataset X_test):

> classifier.predict_proba(X_test_std[:3, :]) where `classifier` is either **K-NN or Logistic Regression**

In the cells below, your must predict the **class-membership probabilities** for each specified **SINGLE ROW OF DATA**

Also, you should recall that the label encoding that we implemented earlier resulted in the mapping:

class membership label indices after encoding

```
0 = 'one' (the Substance is Substance 1)
1 = 'three' (false alarm)
2 = 'two' (The Substance is Substance 2)
```

14) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 10** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

15) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 125** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

16) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 200** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

Build Logistic Regresion Model

17) TO DO: scikit-learn Logistic Regression for this lab

Import and instantiate the *Logistic Regression* Model in SciKit-Learn is in the cell below

```
In [ ]:
```

18) TO DO: Train the model by calling the model's fit function

Now that the model has been instantiated (created) it still needs to be trained (*fitted*) to the training dataset. **Enter your code below to fit the model to the training dataset**

```
In [ ]:
```

19) TO DO: Evaluate the Logistic Regression Model

Use the test set to create the model's predictions. Name the prediction vector **y_pred** as in previous notebooks.

Enter your code in the cell below:

```
In [ ]:
```

20) TO DO: Evaluate the Logistic Regression Model's Performance

Use SciKit Learn's built-in scoring method to evaluate the model's performance accuracy.

Enter your code below:

```
In [ ]:
```

Predicting class-membership probabilities using the Logistic Regression Model

21) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 10** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

22) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 130** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

23) TO DO:class-membership probability

Predict the class membership probability by using the a row with **index = 200** from the X_test_std data. Make sure your print statement uses a complete sentence and be sure to compare your prediction with the correct answer using the corresponding row from the test set labels, y_test.

Enter your code in the cell below.

```
In [ ]:
```

Build Linear Support Vector Classifier Model

24) TO DO:scikit-learn Linear Support Vector Classifier for this lab

Import and instantiate the *Linear Support Vector Classifier* Model in SciKit-Learn is in the cell below

```
In [ ]:
```

25) TO DO: Train the model by calling the model's fit function

Now that the model has been instantiated (created) it still needs to be trained (*fitted*) to the training dataset. **Enter your code below to fit the model to the training dataset**

```
In [ ]:
```

26) TO DO: Evaluate the Linear Suport Vector Classifier Model

Use the test set to create the model's predictions. Name the prediction vector **y_pred** as in previous notebooks.

Enter your code in the cell below:

```
In [ ]:
```

27) TO DO: Evaluate the Linear Suport Vector Model's Performance

Use SciKit Learn's built-in scoring method to evaluate the model's performance accuracy.

Enter your code below:

```
In [ ]:
```

28) TO DO: Using the Predict Method of the Linear Suport Vector Model

Use SciKit Learn's built-in *predict method* to test the model's predictive performance for the first row of data in X_test_std

Enter your code below:

```
In [ ]:
```

29) TO DO:Print the number of misclassifications using numpy

In [ ]:

```

```

Confusion Matrix

Supervised learner models are designed to classify, estimate, and/or predict future outcome. For some applications the desire is to build models showing consistently high predictive accuracy.

Classification Correctness:

Classification correctness is best calculated by presenting previously unseen data in the form of a test set to the model being evaluated. Test set model accuracy can be summarized in a table known as a confusion matrix. To illustrate, let’s suppose we have three possible classes: C1, C2, and C3. A generic confusion matrix for the three class case is shown in Table 1.

- Values along the main diagonal give the total number of correct classifications for each class.

> For example, a value of 15 for C11 means that 15 class C1 test set instances were correctly classified.

- Values other than those on the main diagonal represent classification errors.

> To illustrate, suppose C12 has the value 4. This means that four class C1 instances were incorrectly classified as belonging to class C2. The following three rules may be helpful in analyzing the information in a confusion matrix:

Rules for the Three-Class Confusion Matrix

- Rule 1. Values along the main diagonal represent correct classifications. For the matrix in Table 1,the value C11 represents the total number of class C1 instances correctly classified by the model. A similar statement can be made for the values C22 and C33.
- Rule 2. Values in row Ci represent those instances that belong to class Ci. For example, with i = 2,the instances associated with cells C21, C22, and C23 are all actually members of C2. To find the total number of C2 instances incorrectly classified as members of another class, we compute the sum of C21 and C23.
- Rule 3. Values found in column Ci indicate those instances that have been classified as members of Ci. With i = 2,the instances associated with cells C12, C22, and C32 have been classified as members of class C2.To find the total number of instances incorrectly classified as members of class C2, we compute the sum of C12 and C32.

Table 1 • **A Three-Class Confusion Matrix**

Computed (Predicted)

	c_1	c_2	c_3
c_1	c_{11}	c_{12}	c_{13}
c_2	c_{21}	c_{22}	c_{23}
c_3	c_{31}	c_{32}	c_{33}

30) TO DO: Compute the Confusion Matrix for the Linear Suport Vector Model

Use SciKit Learn's metrics module to compute the model's confusion matrix.

Enter your code below:

In [ ]:

```

```

31)TO DO: Classification Correctness

Based on the output of your **confusion matrix**, what was the total number of *correct* classifications of Substance 2?

Enter your answer as markdown in the cell below:

Type *Markdown* and LaTeX: α^2

32) TO DO: Compute the Classification report for Linear Suport Vector Model

Use SciKit Learn's metrics module to compute the model's classification report.

Enter your code below:

In [ ]:

```

```

33) TO DO: Classification Report

Based on the output of your **classification report**, out of all the times Substance 1 should have been predicted, what percentage of times was it correctly predicted?

Which performance score did you use to evaluate the performance of the model(s) from the confusion matrix /classification report? (HINT: You may need to research the meaning and difference between precision, recall and f1-score)

Enter your answer as markdown in the cell below:

Type *Markdown* and LaTeX: α^2

Serialization

34) TO DO: Model Persistence - Save/Load the trained classifier

To receive full credit for this part you *must test the saved and re-loaded classifiers on an instance of "unknown" data and show that it correctly classifies the instance*. It's ok if you use an instance (sample) that is from the test set as the "unknown" data.

NOTE: Use *either* the Logistic Regression Model or the K-Nearest Neighbors Classifier for this part

Step1: Save the model

Pickle (serialize) and save the trained classifier to a folder

Enter your code below:

In [ ]:

```

```

Step 2: Load the saved model

Load the saved the trained classifier into memory

Enter your code below:

```
In [ ]:
```

Step 3: Test the re-loaded model

Use SciKit Learn's built-in predict method to test the re-loaded model on data from the row of data in X_test_std with index equal to six (6)

Enter your code below:

```
In [ ]:
```

Give a brief (¼ page) report (in the markdown cell below) of your analysis detailing your understanding of the analysis, the machine learning models and their comparative performance, statistical insights, etc. (include reference website links if used).

> Answer in the cell below

Type *Markdown* and LaTeX: α^2

BONUS (15 pts)

It is **your choice** to do the bonus or not. It can only add points to your total course grade. It wont't hurt your grade if you chose NOT to do it. BUT, if you are going to attempt it, do it in the cells below (add cells as necessary). Follow the instructions for the notebook cells layout *exactly* as indicated in the bonus document or you may not receive credit!

```
In [ ]:
```

End of Project

Submit your completed notebook and all associated files, images, datasets, etc. to canvas as a zipped file with properly structured subfolders and NO unnecessary files. You will lose points if you do not adhere to these guidelines.

In [ ]: