University of Sunderland

CET 313 Artificial Intelligence
Workshop 5
Machine Learning Classifiers

Aims of the workshop

In the lecture, we introduced the concept of supervised machine learning classifiers and we looked at some of the most used algorithms such as logistic regression, Support vector Machines, Decision Trees, and Random Forests.

Logistic regression, a powerful algorithm used for binary classification tasks, which models the probability of a certain class or event.
Support Vector Machines (SVM), an algorithm that finds the optimal hyperplane to classify data points.
Decision Trees, which recursively partition the data based on certain features to make decisions, and their ensemble counterparts,
Random Forests, known for their robustness and accuracy in handling complex datasets.

By the end of this tutorial, you will have a foundational understanding of how to apply classifiers to a real-world dataset and evaluate the performance of the model using different error metrics.
Additionally, you will learn how to visually represent the relationship between the chosen feature and the target variable using matplotlib.

Let's begin the tutorial by importing the necessary libraries and loading the dataset.

Feel free to discuss your work with peers, or with any member of the teaching staff.

Reminder

We encourage you to discuss the content of the workshop with the delivery team and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

Exercises herein represent an example of what to do; feel free to expand upon this.

Helpful Resources

Pandas
Panda is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.
10 minutes to pandas — pandas 2.1.2 documentation (pydata.org)

NumPy
NumPy (Numerical Python) is an open-source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems.
NumPy: the absolute basics for beginners — NumPy v1.26 Manual

Matplotlib
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.
Quick start guide — Matplotlib 3.8.0 documentation

Scikit-learn
We will also need scikit-learn library, which is a free machine learning library for Python. It supports both supervised and unsupervised machine learning, providing diverse algorithms for classification, regression, clustering, and dimensionality reduction. The library is built using many libraries you may already be familiar with, such as NumPy and SciPy. It also plays well with other libraries, such as Pandas and Seaborn.
1.2. Linear Models — scikit-learn 1.3.2 documentation

Dataset Hosting Websites:

1. Kaggle:

Kaggle is an online community of data scientists and machine learning practitioners. It offers a platform for data science competitions, hosting datasets and code, and providing a range of educational resources. Kaggle is a valuable resource for data science enthusiasts looking to hone their skills, collaborate with others, and work on real-world data science problems.

https://www.kaggle.com/

2. UCI Machine Learning Repository:

The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators used by the machine learning community for the empirical analysis of machine learning algorithms. It is a widely used repository that provides various datasets for experimental research on machine learning techniques.

https://archive.ics.uci.edu/

3. UCI KDD Archive:

The UCI KDD Archive is a subsection of the UCI Machine Learning Repository, specifically focusing on datasets that are relevant to the KDD (Knowledge Discovery in Databases) process. It includes a variety of datasets that can be used for tasks such as classification, clustering, regression, and other machine learning tasks in the context of data mining and knowledge discovery.

http://kdd.ics.uci.edu/summary.data.application.html

4. UK government, NHS digital, etc.

Environment setup:

In this workshop we will be using few libraries, for the machine learning, thus initially we need to install them. Initial you need to start a new notebook and type the installation commands in code cells, once the installation is complete you need to restart the kernel.

5. Install Pandas

```
!pip install pandas
```

6. Install NumPy

```
!pip install numpy
```

7. Install Matplotlib

```
!pip install matplotlib
```

8. Install scikit-learn

```
!pip install scikit-learn
```

Exercises

You may find it useful to keep track of your answers from workshops in a separate document, especially for any research tasks.
Where questions are asked of you, this is intended to make you think; it would be wise to write down your responses formally.

Exercise 1: Answer the quiz available via canvas.
Exercise 2: Instal the required libraries and its dependences.
Exercise 3: The first step of any machine learning project is to determine a problem to solve and find a relevant dataset that can be used to train the model. In this example, let us use the dataset we presented in the lecture i.e., breast cancer classification.
Exercise 4: Initially we need to import the relevant libraries, as follows:

```
# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, roc_curve, roc_auc_score
```

Please note that we are using the breast cancer dataset, the dataset is hosted on the UCI website (Breast Cancer Wisconsin (Diagnostic) - UCI Machine Learning Repository), and you can also access it via the Sklearn library as follows:

```
# Use the dataset from sklearn library
from sklearn.datasets import load_breast_cancer
```

Exercise 5: To load the dataset, we can easily do the following:

```
# Loading the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target
feature_names = data.feature_names
target_names = data.target_names
```

Exercise 6: Now we can start analysing the dataset to get a better understanding of it. Initially we can assess the classes distribution as follows:

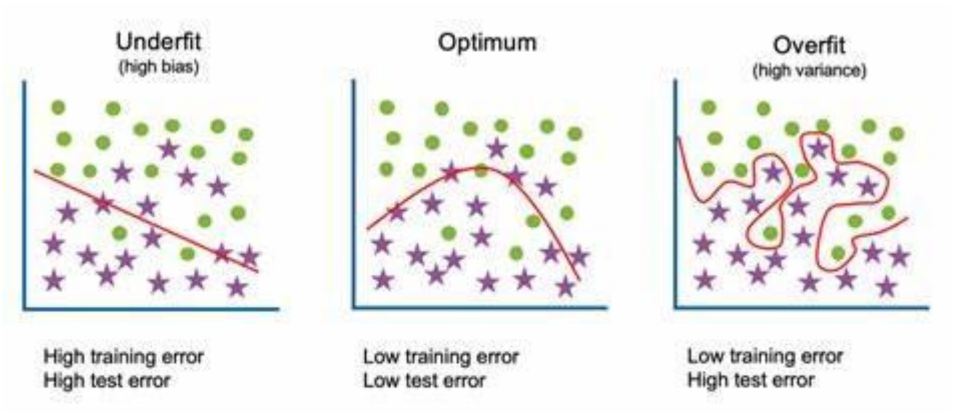
```
# Visualising the classes distribution
plt.figure(figsize=(6, 4))
plt.bar(target_names, np.bincount(y))
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution')
plt.show()
```

[Extra Task] Read about how imbalanced data affects the models performance via: [5. Model Evaluation and Improvement | Introduction to Machine Learning with Python (oreilly.com)]

Exercise 7: To train a model that we can use in real world we need to ensure that it is able to classify correctly, thus it has learned the correct patterns to be able to classify correctly. Thus, we need to test it on a part of the data that it did not see in its training, i.e., the test set, and we can do it as follows:

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

[Extra Task] Read about overfitting via [What is Overfitting? | IBM]



Exercise 8: So far, we have imported the data, and analysed it. Consequently, we can now train the model on the training data as follows:

```
# Fitting logistic regression to the training set
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Exercise 9: Since we now have a trained model, we can use it to predict the classes of the test data as follows:

```
# Predicting the test set results
y_pred = classifier.predict(X_test)
```

Exercise 10: To assess the performance of the classifier we can use the accuracy and other confusion matrix as follows:

```
# Evaluating the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Exercise 11: We can visualise the matrix as follows:

```
# Plotting the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.imshow(conf_matrix, cmap='Blues', interpolation='nearest')
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(2)
plt.xticks(tick_marks, target_names)
plt.yticks(tick_marks, target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Exercise 12: We can also plot the ROC curve using:

```
# Plotting the ROC curve
y_pred_proba = classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.plot(fpr, tpr, label='ROC Curve')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

(Challenge 🧠) [Extra Task] Read about scaling via chapter 3.3.1 of
[3. Preprocessing | Introduction to Machine Learning with Python (oreilly.com)]

To ensure the fairness in feature importance we need to scale that data as follows:

Add the following script to your notebook just after splitting the dataset and rerun the remining part of the code.

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Did the classification result change?

Exercise 13: In real world scenarios we are usually very keen to use the classifiers to identify the most important features which can be important to the medical experts and might lead to new discoveries and a better understanding of the field. Thus, we can visualise the feature importance as follows:

```
# Plotting feature importance
coefficients = classifier.coef_[0]
sorted_indices = np.argsort(coefficients)
plt.barh(range(len(coefficients)), coefficients[sorted_indices],
tick_label=feature_names[sorted_indices])
plt.title('Feature Importance')
plt.xlabel('Coefficients')
plt.ylabel('Feature')
plt.show()
```

Exercise 14: **(Challenge 🧠 🧠 🧠)** Use different classification algorithms such as SVM and Decision Trees and compare the performance. You can import them using the sklearn library. More information can be found at:

1. 1.10. Decision Trees — scikit-learn 1.3.2 documentation
2. sklearn.svm.SVC — scikit-learn 1.3.2 documentation

Exercise 15: **(Challenge 🧠 🧠 🧠)** Now we can try a different dataset and test the whole pipeline on it. The dataset for this exercise is iris dataset Iris - UCI Machine Learning Repository. You can import it as follows:

```
from sklearn.datasets import load_iris
```

Exercise 16: Finally, you can use UCI website to find any dataset for classification and test the algorithms on.

Make sure you upload your notebook with the solutions to your eportfoilio

END OF EXERCISES