University of Sunderland

CET 313 Artificial Intelligence
Workshop 4
Machine Learning

Aims of the workshop

In the lecture, we introduced machine learning, supervised learning and linear regression.

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to improve their performance on a specific task through the use of data, without being explicitly programmed. It involves the study of computer algorithms that can learn from and make predictions or decisions based on data.

Supervised learning is a type of machine learning where the algorithm is trained on a labelled dataset, which means it learns from input-output pairs to predict the output when given new inputs. In supervised learning, the algorithm is presented with a set of input data and corresponding target outputs, and it learns to map the input to the output by generalising patterns from the labelled data.

Linear regression is a fundamental and widely used statistical technique for modelling the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the independent and dependent variables, which allows the algorithm to predict the dependent variable based on the values of the independent variables.

In this tutorial, we will explore how to perform single-variable and multivariable linear regression in Python using the student performance dataset. The dataset, sourced from the UCI Machine Learning Repository, contains various attributes related to student performance, including demographic information and academic indicators.

By the end of this tutorial, you will have a foundational understanding of how to apply linear regression 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.
1.1. 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 a few libraries, for machine learning algorithms, thus initially we will need to instal 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.

1. Instal Pandas

```
!pip install pandas
```

2. Instal NumPy

```
!pip install numpy
```

3. Instal Matplotlib

```
!pip install matplotlib
```

4. Instal 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. The dataset is hosted on the UCI website, and you can access it via the following link:

Cortez,Paulo. (2014). Student Performance. UCI Machine Learning Repository. https://doi.org/10.24432/C5TG7T.

Please download the dataset (in Zip format) and extract all the files in it. Inside the folder you will find another Zip file (named student) also Unzip it. Then copy the student-mat csv file to the same folder as your Jupiter notebook.

Exercise 4: You can open the file using MS excel to check its content.

Exercise 5: To open the dataset, we will be using Pandas library. Thus, initially we need to import the necessary libraries:

```
import pandas as pd
```

Then define the file name, you may need to specify the path it the file is not in the same folder as your notebook, as follows:

```
file_name = 'student-mat.csv'
```

Now you can open the file and print the first 5 rows:

```
df = pd.read_csv(file_name, sep=';')

display(df.head())  # you can also use print instead of display
```

Please note that we used the `';'` because the file separate the values using it, however most common csv files are separated with simple comma (`','`).

Exercise 6: Now we can start analysing the data to get a good understating of it.

```
# Display basic information about the dataset
print("Shape of the DataFrame:")
print(df.shape)

print("\nColumns in the DataFrame:")
print(df.columns)
```

Exercise 7: To get a statistical analysis of the data we can use pandas describe tool, as follows:

```
# Display summary statistics of numerical columns
print("\nSummary Statistics:")
display(df.describe())
```

You can read about the descriptive statistics on the following link:
[Descriptive Statistics | Definitions, Types, Examples (scribbr.com)]()

Exercise 8: To check for any missing data, you can use:

```
# Check for any missing values
print("\nMissing Values:")
print(df.isnull().sum())
```

Exercise 9: To check for duplicated rows, we can use:

```
# Check for any duplicate rows
print("\nDuplicate Rows:")
print(df.duplicated().sum())
```

Exercise 10: To check for column data types, we can use:

```
print("\nData types of the columns:")
print(df.dtypes)
```

Search online for the meaning of these data types.

Exercise 11: We can assess the importance of the feature using the correlation matrix as follows:

```
import matplotlib.pyplot as plt

# Selecting only numerical columns
numerical_columns = df.select_dtypes(include=['int64', 'float64'])

# Calculating correlations
correlations = numerical_columns.corr()

# Creating a basic heatmap to visualise the correlations
plt.figure(figsize=(10, 8))
plt.imshow(correlations, cmap='coolwarm', interpolation='nearest')
plt.colorbar()
plt.xticks(range(len(correlations)), correlations.columns, rotation=90)
plt.yticks(range(len(correlations)), correlations.columns)
plt.title('Correlation Heatmap of Student Performance Dataset')
plt.show()
```

Exercise 12: To understand the features data values.

```
# Generate value counts for categorical variables
print("\nValue Counts for Categorical Variables:")
print(df['sex'].value_counts())
print(df['address'].value_counts())
```

Exercise 13: We can easily visualise the data using matplotlib

```
import matplotlib.pyplot as plt

# Example: Histogram of Age
plt.figure(figsize=(8, 6))
plt.hist(df['age'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Exercise 14: So far, we have read the dataset and done some data analysis on it. Now we can start to build a model. Initially we will need to import the libraries.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Then we need to define the viable we will use to predict the target viable (say the G3 exam grade)

```

# Selecting the feature and target variables
X = df[['G1']]  # Feature variable
y = df['G3']    # Target variable
```

Afterwards, we will split the data to train and test sets (to minimise the risk of overfitting).

```

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

Now we can define the model we will use and train it using the .fit tool:

```
# Creating and fitting the model
model = LinearRegression()
model.fit(X_train, y_train)
```

Now we can use the model to predict the test set.

```
# Predicting the target values
y_pred = model.predict(X_test)
```

Exercise 15: To measure the model performance, we will use 3 matrices i.e., Mean Absolute Error, Mean Squared Error, and Calculating R-squared.

For more information about these metrics you can read about them in this link:
What is Mean Squared Error, Mean Absolute Error, Root Mean Squared Error and R Squared? - Studytonight

```
# Calculating Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)

# Calculating Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculating R-squared
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

Exercise 16: To visualise the performance, we can use:

```

# Plotting the data points and the regression line
plt.scatter(X_test, y_test, color='black', label='Actual data')
plt.plot(X_test, y_pred, color='blue', linewidth=3, label='Regression line')
plt.title('Single Variable Linear Regression')
plt.xlabel('First Test Grade (G1)')
plt.ylabel('Final Grade (G3)')
plt.legend()
plt.show()
```

Exercise 17: **(Challenge 🧠)** Compare the results you got with the results reported in the research paper

Exercise 18: **(Challenge 🧠)** Change the train test split to 70/30 and revaluate the model performance

<u>Exercise 19:</u> **(Challenge 🧠🧠)** Try 3 different independent variables and compare the results. Write a comment on the best variable for the prediction (make sure that the variable you are testing is a numeric one as we cannot use a string as a number).

<u>Exercise 20:</u> **(Challenge 🧠)** Now we will use multiple features to predict the grade and compare the results to the single viable. To do so we need to redefine the X variable as follows. (Note: we cannot plot multidimensional plots, thus you should not plot the data as in exercise 16)

```
#Feature variables
X = df[['G1', 'G2','age', 'traveltime', 'studytime', 'failures']]
```

Look back at the correlation matrix and find the most useful features you may use to get the best results.

<u>Exercise 21:</u> **(Challenge 🧠 🧠 🧠)** Find another dataset on any of the data hosting websites (such as UCI or Kaggle) and download it and repeat all the steps on it.

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

END OF EXERCISES