National University of Computer and Emerging Sciences

Artificial Intelligence Lab

Instructor: Mariam Nasim

Lab contents:

1. **Introduction to pandas**
2. **Linear regression**
3. **KNN**

Pandas**:**

Pandas is a powerful and widely used open-source Python library for **data manipulation and analysis**. It is especially useful for working with **structured data** such as tables, spreadsheets, and time series.

Pandas provides two main data structures:

- **Series** – a one-dimensional labeled array (like a single column of data).
- **DataFrame** – a two-dimensional labeled data structure, similar to a table in Excel or a SQL database.

Key Features:

- Easy data loading from CSV, Excel, SQL, JSON, etc.
- Fast and efficient filtering, sorting, grouping, and reshaping of data.
- Handling of missing data.
- Powerful time series functionality.
- Integration with other libraries like NumPy, Matplotlib, and Scikit-learn.

Importing Data

We can import the libraries or dependency like Pandas using the following command:

```
[ ]  import pandas as pd
```

After the import command, we now have access to a large number of pre-built classes and functions. This assumes the library is installed; in our lab environment all the necessary libraries are installed. One way pandas allows you to work with data is a dataframe. Let's go through the process to go from a comma separated values (**.csv**) file to a dataframe. This variable **csv_path** stores the path of the **.csv** ,that is used as an argument to the **read_csv** function. The result is stored in the object ** df**, this is a common short form used for a variable referring to a Pandas dataframe.

```
[ ]  df=pd.read_csv('top_selling_albums.csv')
```

```
[ ]  type(df)
```

We can use the method **head()** to examine the first five rows of a dataframe:

```
df.head()
```

We can use the method **tail()** to examine the last five rows of a dataframe:

```
[ ]  df.tail()
```

Artist	Album	Released	Length	Genre	Music recording sales (millions)	Claimed sales (millions)	Released	Soundtrack	Rating (friends)
Michael Jackson	Thriller	1982	00:42:19	Pop, rock, R&B	46	65	30-Nov-82		10.0
AC/DC	Back in Black	1980	00:42:11	Hard rock	26.1	50	25-Jul-80		8.5
Pink Floyd	The Dark Side of the Moon	1973	00:42:49	Progressive rock	24.2	45	01-Mar-73		9.5
Whitney Houston	The Bodyguard	1992	00:57:44	Soundtrack/R&B, soul, pop	26.1	50	25-Jul-80	Y	7.0
Meat Loaf	Bat Out of Hell	1977	00:46:33	Hard rock, progressive rock	20.6	43	21-Oct-77		7.0
Eagles	Their Greatest Hits (1971-1975)	1976	00:43:08	Rock, soft rock, folk rock	32.2	42	17-Feb-76		9.5
Bee Gees	Saturday Night Fever	1977	1:15:54	Disco	20.6	40	15-Nov-77	Y	9.0
Fleetwood Mac	Rumours	1977	00:40:01	Soft rock	27.9	40	04-Feb-77		9.5

We can use the attribute **shape** to examine the number of rows and columns of a dataframe:

```
[ ]  df.shape
```

We can access the column "Length" and assign it a new dataframe 'x':

```
[ ]  x=df[['Length']]
     x
```

The process is shown in the figure:

x=df[['Length']]

	Artist	Album	Released	Length	Genre	Music Recording Sales (millions)	Claimed Sales (millions)	Released.1	Soundtrack	Rating
0	Michael Jackson	Thriller	1982	0:42:19	pop, rock, R&B	46.0	65	30-Nov-82	NaN	10.0
1	AC/DC	Back in Black	1980	0:42:11	hard rock	26.1	50	25-Jul-80	NaN	9.5
2	Pink Floyd	The Dark Side of the Moon	1973	0:42:49	progressive rock	24.2	45	01-Mar-73	NaN	9.0
3	Whitney Houston	The Bodyguard	1992	0:57:44	R&B, soul, pop	27.4	44	17-Nov-92	Y	8.5
4	Meat Loaf	Bat Out of Hell	1977	0:46:33	hard rock, progressive rock	20.6	43	21-Oct-77	NaN	8.0
5	Eagles	Their Greatest Hits (1971-1975)	1976	0:43:08	rock, soft rock, folk rock	32.2	42	17-Feb-76	NaN	7.5
6	Bee Gees	Saturday Night Fever	1977	1:15:54	disco	20.6	40	15-Nov-77	Y	7.0
7	Fleetwood Mac	Rumours	1977	0:40:01	soft rock	27.9	40	04-Feb-77	NaN	6.5

x

	Length
0	0:42:19
1	0:42:11
2	0:42:49
3	0:57:44
4	0:46:33
5	0:43:08
6	1:15:54
7	0:40:01

You can do the same thing for multiple columns; we just put the dataframe name, in this case, **df**, and the name of the multiple column headers enclosed in double brackets. The result is a new dataframe comprised of the specified columns:

```
[ ]  y=df[['Length','Artist','Genre']]
     y
```

The process is shown in the figure:

y=df[['Artist' ,'Length', 'Genre ']]

	Artist	Album	Released	Length	Genre	Music Recording Sales (millions)	Claimed Sales (millions)	Released.1	Soundtrack	Rating
0	Michael Jackson	Thriller	1982	0:42:19	pop, rock, R&B	46.0	65	30-Nov-82	NaN	10.0
1	AC/DC	Back in Black	1980	0:42:11	hard rock	26.1	50	25-Jul-80	NaN	9.5
2	Pink Floyd	The Dark Side of the Moon	1973	0:42:49	progressive rock	24.2	45	01-Mar-73	NaN	9.0
3	Whitney Houston	The Bodyguard	1992	0:57:44	R&B, soul, pop	27.4	44	17-Nov-92	Y	8.5
4	Meat Loaf	Bat Out of Hell	1977	0:46:33	hard rock, progressive rock	20.6	43	21-Oct-77	NaN	8.0
5	Eagles	Their Greatest Hits (1971-1975)	1976	0:43:08	rock, soft rock, folk rock	32.2	42	17-Feb-76	NaN	7.5
6	Bee Gees	Saturday Night Fever	1977	1:15:54	disco	20.6	40	15-Nov-77	Y	7.0
7	Fleetwood Mac	Rumours	1977	0:40:01	soft rock	27.9	40	04-Feb-77	NaN	6.5

y

	Artist	Length	Genre
0	Michael Jackson	0:42:19	pop, rock, R&B
1	AC/DC	0:42:11	hard rock
2	Pink Floyd	0:42:49	progressive rock
3	Whitney Houston	0:57:44	R&B, soul, pop
4	Meat Loaf	0:46:33	hard rock, progressive rock
5	Eagles	0:43:08	rock, soft rock, folk rock
6	Bee Gees	1:15:54	disco
7	Fleetwood Mac	0:40:01	soft rock

Adding Column

```
[ ] df['New Artist'] = df['Artist']
```

```
[ ] df.head()
```

Droping Column

```
[ ] df.drop(['New Artist'], axis=1, inplace=True)
```

```
[ ] df.head()
```

Object Type of each column

```
[ ] df.dtypes
```

Null values check in Data Frame

```
[ ] df.isnull()
```

```
[ ] df.isnull().sum()
```

Summary Statistics

```
[ ] df.describe()
```

```
[ ] df.describe(include='all')
```

Querying a dataframe

Querying a database means finding some values based on certain conditions. For example you want to find out the albums having rating greater and equal to 9

```
[ ] soundtracks= df[df['Rating']>=9.0]
```

```
[ ] soundtracks
```

Notice that in the above result all the columns are displayed. If you want to access a specific column, then you can use **loc** for that purpose.

```
[ ] soundtracks_album= df.loc[df['Rating']>=9.0, ['Album']]
```

```
[ ] soundtracks_album
```

b. Handling Missing Values

python Copy Edit

```python
df.isnull().sum()           # Count missing values
df.dropna()                 # Drop rows with missing values
df.fillna(0)                # Fill missing values with 0
df['Age'].fillna(df['Age'].mean())  # Fill with mean
```

c. Encoding Categorical Data

python Copy Edit

```python
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
df = pd.get_dummies(df, columns=['Category'])  # One-hot encoding
```

d. Normalizing or Scaling Data

python Copy Edit

```python
# Min-Max normalization
df['Score'] = (df['Score'] - df['Score'].min()) / (df['Score'].max() - df['Score'].min())
```

More details can be studied here: https://www.w3schools.com/python/pandas/default.asp

Regression:

A general flow of implementing regression is given below:

1. Import Libraries

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

2. Load the Dataset

```python
df = pd.read_csv('your_dataset.csv')  # Or any data source
print(df.head())
```

3. Explore and Preprocess the Data

- Handle missing values
- Convert categorical to numeric if needed
- Drop irrelevant columns

```python
df = df.dropna()  # or df.fillna()
# df['Category'] = df['Category'].map({'A': 0, 'B': 1})
```

🔷 4. Define Features and Target

python

```python
X = df[['feature1', 'feature2']]  # Independent variables
y = df['target']                  # Dependent variable
```

🔷 5. Split into Train and Test Sets

python Copy Edit

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

🔷 6. Train the Regression Model

python

```python
model = LinearRegression()
model.fit(X_train, y_train)
```

🔷 7. Make Predictions

python

```python
y_pred = model.predict(X_test)
```

🔷 8. Evaluate the Model

python

```python
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R-squared Score:", r2_score(y_test, y_pred))
```

KNN:

Details can be studied here: https://www.geeksforgeeks.org/k-nearest-neighbours/

Lab Tasks:

Task 1: Regression

1. Download advertising dataset from this link: https://www.kaggle.com/datasets/ashydv/advertising-dataset/data
2. Load data into a pandas dataframe and check its summary statistics and object type of each column.
3. Perform data cleaning
   - check null values, outlier analysis
4. Perform Exploratory data analysis
   - see how Sales are related with other variables using scatter plot
   - see correlation between different variables using heatmap
5. Build a simple linear regression model to predict sales. Split data into test (20%) and train(80%)
6. Evaluate the model (root mean squared error)
7. Visualize results using scatter plot

Task 2: KNN

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. It can be downloaded in csv from here:
https://www.kaggle.com/datasets/oddrationale/mnist-in-csv

1. Load the data
2. Visualize data (display 10 image samples from the training data , 1 from each class)
3. Train a KNN classifier to classify each image into one of the classes (0-9)
4. Evaluate the model
5. Build confusion matrix
6. Examine the classification report of our KNN model(precision, recall, f1-score, support)

You can use libraries like Scikit-learn.