NAiSS Advanced MLB

Session #2

January 29th, 2025

Data Preprocessing & More

Use Google + ChatGPT (or Deepseek) when you have small questions. It’s a skill and makes learning faster.

Before we start

1. What is Data Preprocessing?
2. Data Cleaning
3. Data Transformation
4. Data Reduction
5. Data Splitting
6. Git 101
7. Building a Model

Index & Content

It is the process of preparing and cleaning raw data before it is used for making predictions.

Think of it like preparing the ingredients before cooking a meal, the wrong amount or too much mess will ruin the end result.

What is Data Preprocessing?

1.

Data Cleaning

2.

Data Transformation

3.

Data Reduction

4.

Data Splitting

General 4 step Process

The process of fixing or removing incorrect, incomplete, or irrelevant data.

Bad Data = Bad Results

Common Issues:

- Missing Values (Blank Cells)
- Duplicate Datapoints/Rows
- Incorrect Values (ex. Negative age)

Data Cleaning

- **Missing Values** - Fill them with average values or remove them.
- **Duplicates** - Keep only one copy of each unique entry.
- **Incorrect Values** - Replace or remove data points that are obviously incorrect.

If the data doesn’t make sense, fix it or remove it.

Data Cleaning Simple Solutions

The process of changing the structure of data to make it suitable for making predictions.

Transforming data is like resizing an image, sometimes you need to adjust it to get the more accurate picture.

Basic Techniques:

- Scaling/Normalization
- Encoding Categorical Data

Data Transformation

- **Scaling/Normalization** - Adjusting the range of numerical data to be between 0 and 1 so some features do not overpower others.

If you have two features, one regarding house prices and the other with age. At face value, the model might interpret that the house prices are more important since the numbers are larger.

- **Encoding Categorical Data** - Turning categories (like “Yes” and “No” or “Coke”, “Pepsi”, or “Sprite”) into numbers.

This makes it easier for the model to process, as they become numerical.

Data Transformation Techniques

The process of reducing the size of the dataset while retaining all of the important information.

Large datasets can be slower to process and harder to work with. By reducing the size, we can get past this and also avoid overfitting.

Basic Techniques:

- Feature Selection
- Data Sampling
- Dimensionality Reduction (Not as basic)

Data Reduction

- **Feature Selection** - choosing the most important features, either manually or using model-based methods.

If you are trying to predict the price of eggs in the next month, you most likely would not need a feature in your dataset for the shoe size of dogs, as there is no relevant relationship to the target variable.

- **Data Sampling** - select a representative subset of the data (either random or stratified) to reduce size.

This can either reduce size for quicker analysis or ensure that important groups are fairly represented in the sample.

Data Reduction Techniques

The process of dividing your dataset into subsets to train, test, and possibly validate your machine learning model.

It is important to ensure your model can generalize well to new, unseen data.

Think of it like **studying for a test**:

- You practice with study materials. (training data)
- You take a mock exam to see how well you actually understand the material. (testing data)
- Finally, you take the real exam (model deployment)

Data Splitting

Consists of 70-80% of the original dataset.

This set is used to train the model.

Training Set

Consists of the other 20-30% of the original dataset.

This set is used to evaluate the model’s performance on new data.

Testing Set

Git 101 (for those College of IST'ers)

Git is a version control system that helps you track changes in your code and collaborate with others.

Version control is used in virtually every company/team where software and tech is involved. It’s very important.

Today, we just want to use it to track changes in code.

What is Git?

You can run these commands in your terminal after you have cloned the project repository onto your computer.

- **git add <file to add>** - stages changes to Git’s tracking
- **git commit -m “message”** - commits your changes with a message of what was changed
- **git push** - Pushes changes to remote repository (GitHub)
- **git pull** - Fetches latest changes from remote repository (GitHub)

Basic Commands

A Simple Model

After you have prepared your data, it’s time to choose a model, train it, then evaluate it.

There are two main types of models:

- Classification Models (predicting categories)
- Regression Models (predicting continuous values)

For today, we will go over a simple classification model, **Logistic Regression**.

Building a Model

Logistic Regression is a machine learning algorithm used for binary classification problems, where the goal is to predict one of two outcomes (yes or no / true or false)

It calculates a probability that an input belongs to a certain class, you can then use a threshold to determine if the input is in one of two classes.

Logistic Regression

Pandas is a powerful Python library used for data manipulation and analysis, providing a data structure (**DataFrames**)to handle large datasets efficiently. Read documentation / Google specific commands when needed, but here are a few common useful commands:

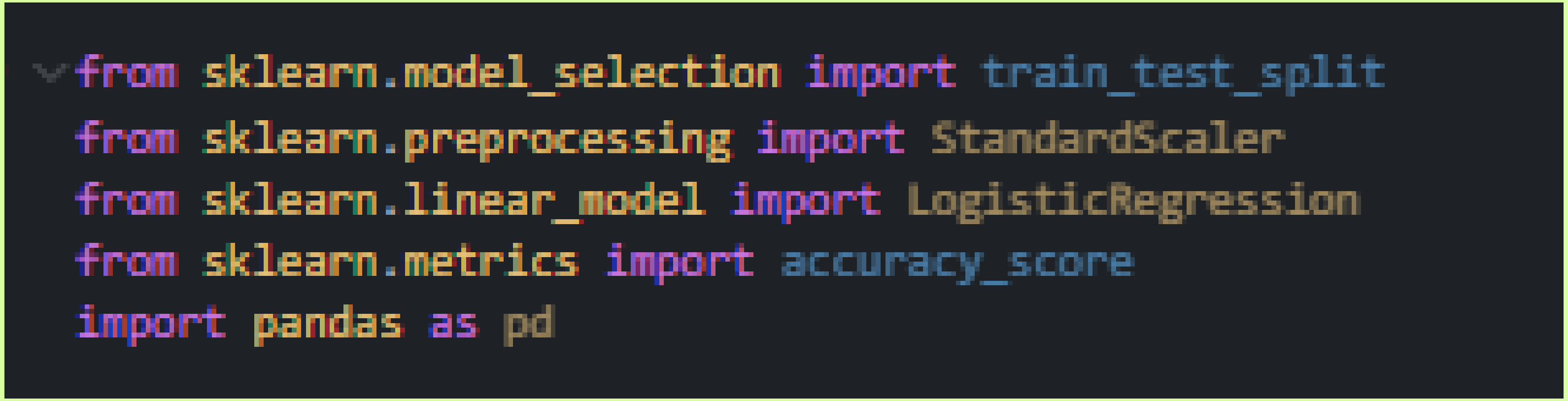
- df.head() - view the first few rows of a dataframe
- df.dropna() - remove rows with missing values
- df.fillna() - fill missing values with a specific value
- df.drop() - drop specified rows / columns
- df.drop_duplicates() - remove duplicate rows

Pandas

Scikit-learn is a popular python library that provides the toolbox for data analysis and modeling. It works well with NumPy and pandas.

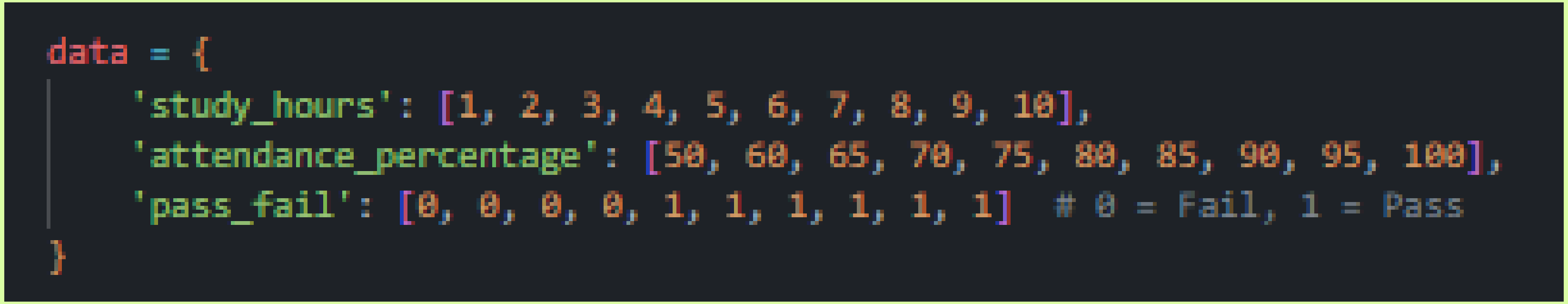
You can use these libraries to set up a model to make predictions and classifications. We will go through a really brief example and then it is up to you.

Scikit-learn



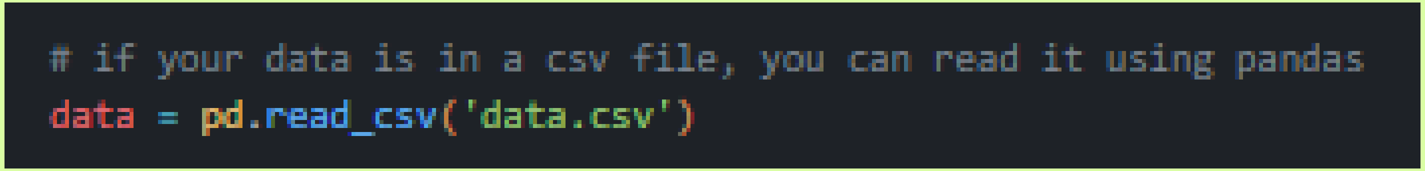
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
```

Importing Libraries



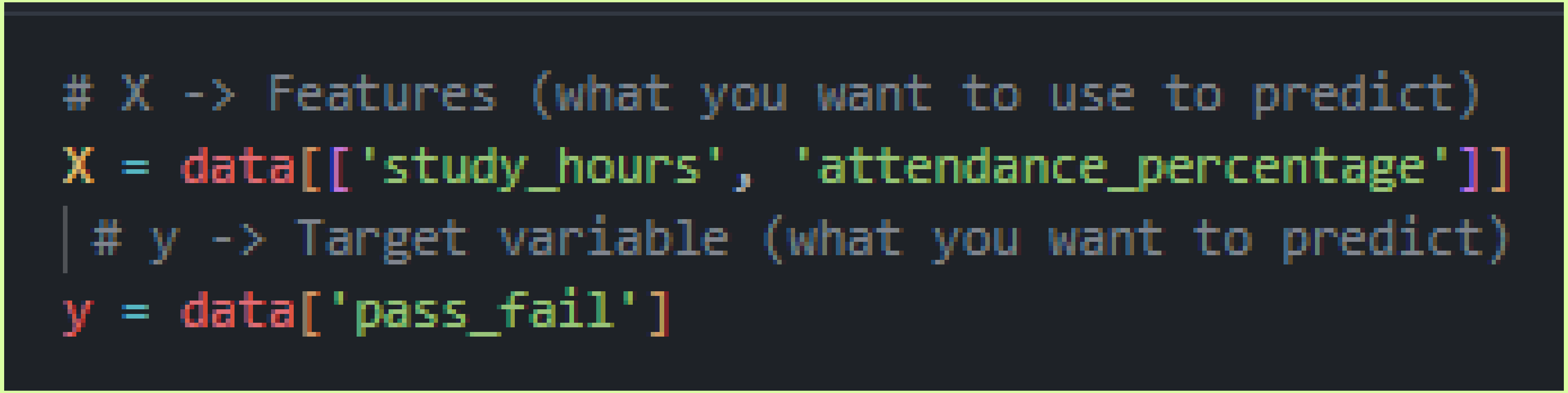
```
data = {
    'study_hours': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'attendance_percentage': [50, 60, 65, 70, 75, 80, 85, 90, 95, 100],
    'pass_fail': [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]  # 0 = Fail, 1 = Pass
}
```

OR



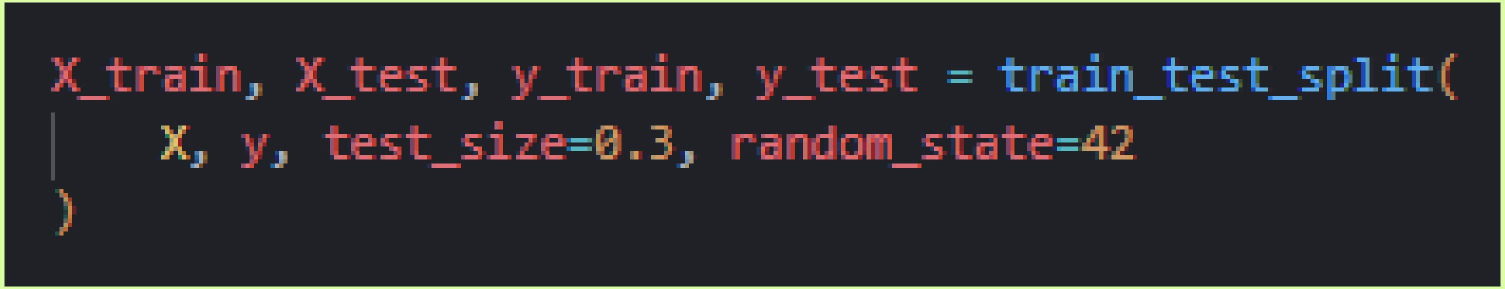
```
# if your data is in a csv file, you can read it using pandas
data = pd.read_csv('data.csv')
```

Importing Data



```
# X -> Features (what you want to use to predict)
X = data[['study_hours', 'attendance_percentage']]
 # y -> Target variable (what you want to predict)
y = data['pass_fail']
```

Splitting into Features and Target Variables



```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

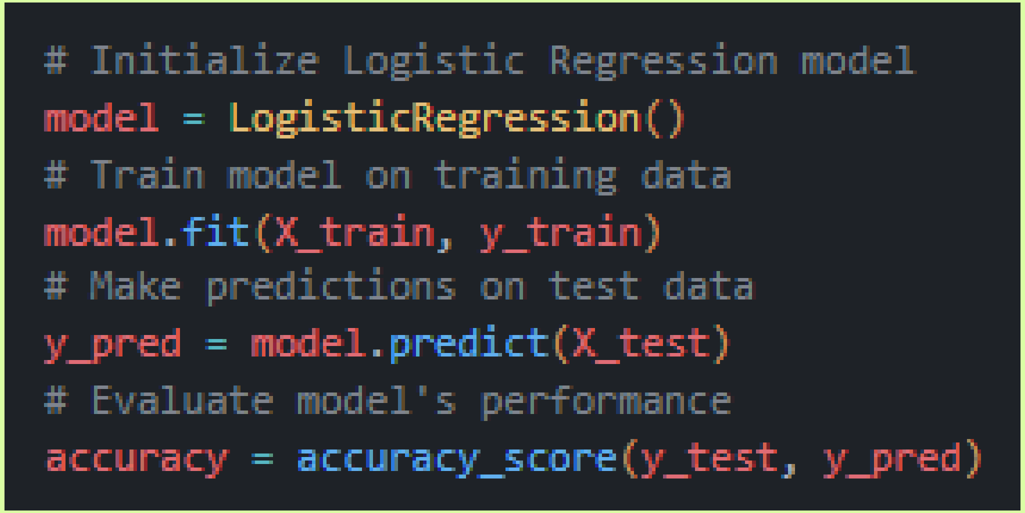
Splitting into Train and Test Data

This is where you would do any data preprocessing steps that you feel are necessary.

Data preprocessing before splitting can lead to **data leakage**, and information from the test set could technically influence the training process.

- Imagine scaling the data before splitting, the testing data then learns the scale of the test data as well.

Data Preprocessing


```
# Initialize Logistic Regression model
model = LogisticRegression()
# Train model on training data
model.fit(X_train, y_train)
# Make predictions on test data
y_pred = model.predict(X_test)
# Evaluate model's performance
accuracy = accuracy_score(y_test, y_pred)
```

Run the Model

Thank you for coming - Zach Walnock