COSC 2753 | Machine Learning

Week 3 Lab Exercises: **Training a Regression Model**

Introduction

Last week we learned how to read the data and do exploratory data analysis (EDA). The next step in a typical machine learning pipeline is to split data and transform so that we can feed the data to a learning algorithm.

The lab assumes that you have completed `Week 02 lab: Reading data & Exploratory Data Analysis (EDA)`. If you haven't yet, please do so before attempting this lab.

The lab can be executed on either your own machine (with anaconda installation) or computer lab.

- Please refer canvas for instructions on installing anaconda python

Objective

- Continue to familiarise with Python and other ML packages
- Learn to split the data to training/validation and test sets
- Important considetations in splitting the data
- Learn to train a model for regression and complete the introduction to model development pipeline.

Dataset

We examine two regression based datasets in this lab. The first one is to do with house prices, some factors associated with the prices and trying to predict house prices. The second dataset is predicting the amount of share bikes hired every day in Washington D.C., USA, based on time of the year, day of the week and weather factors. These datasets are available in `housing.data.csv` and `bikeShareDay.csv` in the code repository.

First, ensure the two data files are located within the Jupyter workspace.

- If you are on the local machine copy the two data data directories (`BostonHousingPrice`,`Bike-Sharing-Dataset`) to your current folder.

In this course we mostly use datasets that are collected by a third party. If you are interested in collecting your own data for your project, some useful information can be found at: [Introduction to Constructing Your Dataset]

Problem Formulation

The first step in developing a model is to formulate the problem in a way that we can apply machine learning. To reiterate, the `task` in the Boston house price dataset is to predict the house price (`MEDV`), using some attributes of the house and neighbourhood.

 Observe the data and see if there is a pattern that would allow us to predict the house price using the attributes given? You can use the observations from the EDA for this.

 What category does the task belong to?

> ✔ **Task category:**
>
> - supervised, univariate/multivariate regression

> - We should use the insights gained from observing the data (EDA) in selecting the performance measure. e.g. are there outliers in target?

Data Splitting

As we have discussed in the lecture, in supervised learning we are interested in learning a model using our dataset that can predict the target value for unseen data (Not in the training set). This is called **generalization**. How can we test if the model we developed with our training data would generalize? One approach we can use is to **hold some data from the training process** (Hypothetical unseen data). This hold out data subset (split) is called the `"Test set"` and the remaining data is called the `”Training set"`. The training set may be further subdivided, but

more on this later in the regularization lecture. We can use the "Test set" at the end of the development phase to test our model and see if it generalizes.

- **Training set:** Is applied to train, or fit, your model. For example, you use the training set to find the optimal weights, or coefficients, for linear regression, logistic regression, or neural networks.
- **Test set:** Needed for an unbiased evaluation of the final model.

> ⚠ **Warning: The test set should be independent and identically distributed with respect to the training data**
>
> - Should make sure that there is no leakage between the two sets (overlapped train and test instances). This will give unrealistically high performance metric values for your model. e.g. In house price prediction, there may be a house that was sold multiple times and, you might include some instances of this house in the train set and some in the test set. This will result in data leakage.
> - There should be no underlying differences between the two distributions. In other words the characteristics of the test set should not be different to that of the train set. For example all the houses sold in winter in train set and all the houses sold on summer in another set (generally, there is a difference in house prices sold in winter vs summer).
> - More on this in the lectures.

> ⚠ **Warning: The test data should NOT be used for any aspect of the model development process (training).**
> This includes hyper parameter tuning and model selection (a separate validation set should be used for them).

Random splitting

In machine learning, the most common approach taken to split the dataset in to do random sampling (random split). In random sampling we allocate some data instances (selected randomly) to train set and the other instances to test set. One key configuration parameter in this process: what should be the size of the train set and test set respectively. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project's objectives with considerations that include:

- Computational cost in training/evaluation the model.
- Training set/Test set representativeness.

Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Lets first load the house price dataset.
Use the knowledge from the last week to load the dataset into a pandas dataframe named `bostonHouseFrame`.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

## TODO
bostonHouseFrame = pd.read_csv("housing.data.csv", delimiter="\s+")
print(bostonHouseFrame)

```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273.0
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273.0
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273.0
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273.0
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273.0

     PTRATIO       B  LSTAT  MEDV
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
```

```
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

The scikit-learn Python machine learning library provides an implementation of the train-test splitting via function `train_test_split()`. Lets use this to randomly split our data to 80% train set and 20% test set.

```
from sklearn.model_selection import train_test_split

with pd.option_context('mode.chained_assignment', None):
    bostonHouseTrainFrame, bostonHouseTestFrame = train_test_split(bostonHouseFrame, test_size=0.2, sh

```

```
print("Nunber of instances in the original dataset is {}. After spliting Train has {} instances and te
      .format(bostonHouseFrame.shape[0], bostonHouseTrainFrame.shape[0], bostonHouseTestFrame.shape[0]
```

```
Nunber of instances in the original dataset is 506. After spliting Train has 404 instances and test has 102 instance
```

Checking your splits

As discussed, random splitting may lead to leakage (two splits are not independent). We need to understand the dataset to make sure there are no hidden sources of leakage in the data. This is one place we can use the knowledge we gained through the EDA.

> ☞ **Task: Use the EDA observations from last week to see if there is any issue with the random splitting process.**

We can use histogram plots to see if the two partitions (splits) are identically distributed.

 Use the knowledge from last week to plot the histograms for each attribute for the two splits. Use different colors for test vs train

[ ] *4 cells hidden*

Univariate Regression

We will first study how to do univariate regression.

If you recall from the last lab, we found that possibly the 'RM' (number of rooms) and 'LSTAT' (unsure) variables seem to have a linear relationship with the house price ('MEDV'). Hence, we will try these variables as the independent variable to predict the house price, i.e., the dependent variable.

[ ] *51 cells hidden*

Exercise: Work on the Bike Share Data

> ☞ **Task: Do the linear regression on the Bike Share Data.**
> Now you seen how to do this task for the house price dataset. Repeat the same process for the Daily Bike Share rental data.

[ ] *8 cells hidden*

Some other data pre-processing steps

> **☞ Goal: Do some pre-process steps before fitting the data for training

[ ] *15 cells hidden*