TASK 1A

This task is about using **cross-validation** for **ridge regression**. Remember that ridge regression can be formulated as the following optimization problem:

$$\min_{w} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||_2^2$$ $\min_w \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_2^2$

where y_i is the label of the ith datapoint, x_i is the vector of features for the ith datapoint and λ is a hyperparameter.
Consider the following set of regularization parameters:

λ_1	λ_2	λ_3	λ_4	λ_5
0.1	1	10	100	200

Your task is to perform **10-fold cross-validation** with **ridge regression** for each value of λ given above and report the **Root Mean Squared Error (RMSE) averaged over the 10 test folds**. In other words, for each λ, you should train a ridge regression 10 times leaving out a different fold each time, and report the average of the RMSEs on the left-out folds. More background on K-fold cross-validation and some hints on useful library functions can be found here in the scikit-learn user guide. You can assume that each datapoint in your dataset is sampled independently from the same distribution (iid), so section 3.1.2.1 of the user guide should be particularly relevant.
The RMSE is defined as

$$\mathrm{RMSE} = \sqrt{\boxed{1/n \sum_{i=1}(y_i - \hat{y}_i)}}$$

where y_i are the ground truth labels and $\hat{y}_i$ estimations made by your regressor. You should perform the linear regression on the original features, i.e., you **should not perform any feature transformation or scaling**. Note that the goal here is to test your understanding by accurately performing the described cross-validation of the machine learning method we've described (linear regression with ridge). This is different to future tasks, where the goal will normally be to have the most accurate predictions.

DATA DESCRIPTION

In the handout for this project, you will find the the following files:

- **train.csv** - the training set
- **sample.csv** - a sample submission file in the correct format
- **template_solution.py** - a template file that will guide you through the implementation of the solution
- **template_solution.ipynb** - a template file in jupyter notebook format that will guide you through the implementation of the solution

You are free to use either jupyter notebook or the .py template file.

Each line in train.csv represents one datapoint by its label y, and its features x1 to x13:

```
y,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13
22.6,0.06724,0.0,3.24,0.0,0.46,6.333,17.2,5.2146,4.0,430.0,16.9,375.21,7.34
...
```

For your convenience, we further provide a sample submission file:

```
13.1
9.6
6.0
14.5
22
```

We provide a template solution file that suggests a structure for how you can solve the task, by filling in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier. While it is not mandatory, we encourage you to implement ridge regression from scratch, to get a deeper understanding of the course material.

SUBMISSION FORMAT

The submission file format should be identical to the format of sample.csv, i.e., it should have exactly 5 lines, each containing a floating point number that represents the average RMSE scores obtained for λ_1,λ_2,λ_3,λ_4,λ_5 (in this order).
Please keep in mind that, as a group, you have a limited number of submissions as stated on the submissions page.

TASK 1B

This task is about **linear regression**: given an input vector **x**, your goal is to predict a value **y** as a **linear** function of a set of feature transformations, $\phi(\mathrm{x})\phi(x)$.

DATA DESCRIPTION

In the handout for this project, you will find the the following files:

- **train.csv** - the training set
- **sample.csv** - a sample submission file in the correct format
- **template_solution.py** - a template file that will guide you through the implementation of the solution
- **template_solution.ipynb** - a template file in jupyter notebook format that will guide you through the implementation of the solution

You are free to use either jupyter notebook or the .py template file.

Each line in train.csv represents one data instance by an id, its label y, and its features x1-5:

```
Id,y,x1,x2,x3,x4,x5
0,3.5796196352704994,0.019999999999999907,0.04999999999999993,-0.09000000000000008
,-0.43000000000000005,-0.08000000000000007
...
```

FEATURES DESCRIPTION

You are required to use the following features (in the following order) to make your predictions:

- Linear $\phi_1(x)=x_1$, $\phi_2(x)=x_2$, $\phi_3(x)=x_3$, $\phi_4(x)=x_4$, $\phi_5(x)=x_5$
- Quadratic $\phi_6(x)=x_1^2$, $\phi_7(x)=x_2^2$, $\phi_8(x)=x_3^2$, $\phi_9(x)=x_4^2$, $\phi_{10}(x)=x_5^2$,
- Exponential $\phi_{11}(x)=e^{x_1}$, $\phi_{12}(x)=e^{x_2}$, $\phi_{13}(x)=e^{x_3}$, $\phi_{14}(x)=e^{x_4}$, $\phi_{15}(x)=e^{x_5}$
- Cosine $\phi_{16}(x)=\cos(x_1)$, $\phi_{17}(x)=\cos(x_2)$, $\phi_{18}(x)=\cos(x_3)$, $\phi_{19}(x)=\cos(x_4)$, $\phi_{20}(x)=\cos(x_5)$
- Constant $\phi_{21}(x)=1$

where we indicate the whole input vector with **x** and we use x_i to denote its ith component.

Your predictions are calculated as a linear function of the features above according to the following formula:

$\hat{y}=w_1\phi_1(x)+w_2\phi_2(x)+\cdots w_{21}\phi_{21}(x)$

We provide a template solution file that suggests a structure for how you can solve the task, by filing in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier. You are also encouraged (but not required) to implement regression solutions from scratch, for a deeper understanding of the course material.

TASK 2

INTRODUCTION

Applying ML approaches to real-world problems requires some extra steps in handling specific data artifacts. Some common challenges you may face are missing values, an imbalance of the labels, or noise in the data distribution. These can be addressed through a specific choice of pre-processing and/or model components.

In this task, as an illustration of a real-world problem, you are asked to predict the electricity prices in Switzerland given price information of some other countries and additional features. You will encounter typical ML workflow challenges of missing features and low predictivity in this task.

The following sections provide more details on the dataset, submission and evaluation.

DATA DESCRIPTION

In the handout for this project, you will find the the following files:

- **train.csv** - the training set
- **test.csv** - the test set file to make predictions on
- **sample.csv** - a sample submission file in the correct format
- **template_solution.py** - a template file that will guide you through the implementation of the solution
- **template_solution.ipynb** - a template file in jupyter notebook format that will guide you through the implementation of the solution

You are free to use either jupyter notebook or the .py template file.

Each line in **train.csv** represents one data point and consists of the following structure:

```
season,price_AUS,price_CHF,price_CZE,price_GER,price_ESP,price_FRA,price_UK,price_
ITA,price_POL,price_SVK

spring,,9.644027877268496,-1.6862480951361345,-1.7480763846576997,-3.6660054011856
53,,-1.822719793809122,-3.931031226630509,,-3.238196806151894
...
```

As mentioned in the introduction, we are interested in predicting the electricity (log) prices in Switzerland (corresponds to price_CHF column), given the prices in some other countries (corresponds to all columns starting with "price_" except price_CHF), and additional features (corresponds to season). Note that there might be missing values in the price_CHF column as well.
test.csv is the file you submit predictions on. The file consists of the following structure:

```
season,price_AUS,price_CZE,price_GER,price_ESP,price_FRA,price_UK,price_ITA,price_
POL,price_SVK

spring,,0.4729846640395274,0.7079565159369821,,-1.1364407652408537,-0.596702855735
1153,,3.298693142239604,1.9218859561660853
...
```

For your convenience, a sample solution file **sample.csv** is provided with the following structure:

```
price_CHF
0.635389524381449
0.635389524381449
0.635389524381449
0.635389524381449
```

template_solution.py provides a starting template structure for how you can solve the task, by filling in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier.

MODELLING TIPS

DATA IMPUTATION

Missing data is a commonly encountered artifact in several machine learning tasks. Typical imputation strategies to deal with missing data include:

- Discarding rows or columns with missing data
- Replacing missing values with the corresponding mean or median
- Advanced imputation strategies based on iterative model fitting.

Additional resources on data imputation can be found in this kaggle notebook or sklearn website.

HANDLING CATEGORICAL (NON-NUMERIC) DATA

Some data in this task is categorical (non-numeric). This is a common challenge in machine learning. Some strategies on how to handle categorical data can be found in this kaggle notebook or sklearn website.

KERNELIZED REGRESSION MODELS

You saw kernelized estimators in the lecture notes. In this task you might find them useful. The core of the challenge is to pick the right kernel for the regression. Commonly used kernels are the linear (or dot product) kernel, squared exponential (or RBF) kernel, polynomial, Matern, and RationalQuadratic kernels among many others. A probabilistic (Bayesian) equivalent of kernelized ridge regression goes by the name Gaussian processes. It provides a principled modelling paradigm with uncertainty estimates. The uncertainty component is not important for this task, however a lot of machine learning software packages implement this method in a very efficient manner and its mean prediction does the same as kernelized ridge regression. The following code block gets you started with gaussian processes in sklearn (more resources can be found here)

```
from sklearn.gaussian_process import GaussianProcessRegressor

from sklearn.gaussian_process.kernels import DotProduct, RBF, Matern, RationalQuad
ratic

gpr = GaussianProcessRegressor(kernel=DotProduct())

gpr.fit(X_train, y_train)
```

Finding the right kernel can be done in multiple ways as you saw in the lectures:

- Using a validation set
- Cross-validation

- Maximizing the evidence of a Bayesian model. In this case, we are maximizing the total probability of the data given its generative model. This is also referred to as Bayesian model selection. More information can be found here: Bayesian model selection - lecture notes.

All the above methods are implemented in scikit-learn. Note that Gaussian processes implement Bayesian model selection automatically.

SUBMISSION FORMAT

The submission file format should be the same as that of **sample.csv**, i.e. the file must contain a column with name price_CHF, and each row is the corresponding prediction. Please ensure that the number of predictions in your submitted file are the same as number of data points in the **test.csv** file.

Furthermore, please keep in mind that as a group, you have a limited number of submissions as stated on the submissions page.

EVALUATION

We are interested in an accurate prediction of electricity prices in Switzerland. As mentioned before, this corresponds to the price_CHF column in your submitted predictions. Your submitted predictions will be evaluated by the *R-squared* (R_2R2) metric to the true prices (here, a higher score is better).
$R_2(y_*,y)=1-\sum_{Ni=1}(y_{*i}-y_i)_2\sum_{Ni=1}(y_{*i}-\bar{y})_2$
where $\bar{y}$ denotes the average of true values $\{y_{*i}\}\{yi*\}$.
To calculate the R_2 score, we use the scikit-learn implementation:

```
from sklearn.metrics import r2_score

r2_score(y_true, y_pred, squared=False)
```

TASK 3

In this task, you will make decisions on food taste similarity based on images and human judgements.We provide you with a dataset of images of 10000 dishes, a sample of which is shown below.

We also provide you with a set of triplets (A, B, C) representing human annotations: the human annotator judged that the taste of dish A is more similar to the taste of dish B than to the taste of dish C. A sample of such triplets is shown below.

You task is to predict for unseen triplets (A, B, C) whether dish A is more similar in taste to B or C.

DATA DESCRIPTION

In the handout for this project, you will find the the following files:

- **dataset** - folder containing the dish images
- **train_triplets.txt** - contains the training triplets. The entries of each triplet denote file names. For example, the triplet "00723 00478 02630" denotes that the dish in image "00723.jpg" is more similar in taste to the dish in image "00478.jpg" than to the dish in image "02630.jpg" according to a human annotator.
- **test_triplets.txt** - the triplets you should make predictions for
- **sample.txt** - a sample submission file
- **template_solution.py** - a template file that will guide you through the implementation of the solution
- **template_solution.ipynb** - a template file in jupyter notebook format that will guide you through the implementation of the solution

You are free to use either jupyter notebook or the .py template file.

template_solution.py provides a starting template structure for how you can solve the task, by filling in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier. The template solution uses the PyTorch deep learning framework. We advise you to use PyTorch for this task and on the following links you can get started with PyTorch:

- 60 Minute Blitz

- PyTorch with Examples
- Tensors

The computational demands for this task are higher than the previous tasks. Modern laptops with sufficient RAM (~16-20 GB) should be able to handle the task. If you are on Ubuntu and you find yourself with insufficient RAM, you can consider increasing swap memory. You can find a guide on how to do that here. If you still find yourself requiring more computational power, you can use the Euler cluster. Though you should also consider the setup time when deciding if you need it. Below you can find a guide on how to set up a GPU-enabled environment on Euler.
Your task is the following: for each triplet (A, B, C) in **test_triplets.txt** you should predict 0 or 1 as follows:

- 1 if the dish in image A.jpg is closer in taste to the dish in image B.jpg than to the dish in C.jpg
- 0 if the dish in image A.jpg is closer in taste to the dish in image C.jpg than to the dish in B.jpg.

Using pretrained models is very common in current deep learning practice and a useful tool to have in your practical toolbox. PyTorch provides a number of pretrained models that you can use. You can find a list of the available models and how to use them here. To learn more about transfer learning and how to use pretrained models you can check the following link: Transfer Learning Tutorial.

EVALUATION

Your submission will be compared to judgements produced by human annotators and the simple accuracy of your predictions will be measured:

$$\text{Accuracy}=\text{Number of correct predictions}/\text{Number of total predictions}$$

TASK 4

GENERATING SCORES FROM USER REVIEWS

Welcome to task #4, the last project of this year’s Introduction to Machine Learning course at ETH!

When customers shop online, they can leave reviews for the purchased product. In this task, given a dataset containing the review's title and its content, you will try to infer the score of the product, which is a continuous value between 0 and 10.

DATA DESCRIPTION

You can download the project material here:

In the handout, you will find the following material:

- **train.csv** - a csv containing the columns "title", "sentence", and "score". The score is always within the range [0, 10].
- **test_no_score.csv** - csv containing the columns "title" and "sentence".
- **sample.txt** - a sample submission file.
- **template_solution.py** - a template file that will guide you through the implementation of the solution.
- **template_solution.ipynb** - the template solution in an interactive notebook form.

template_solution.py provides a starting template structure for how you can solve the task, by filling in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier. The template solution uses the PyTorch deep learning framework.

SUGGESTED APPROACH

The recommended approach is to use the transformers library, which contains the necessary network architectures and pretrained transformers. It is recommended to look into the following transformers:

- DistilBERT
- ALBERT
- GPT-2

Note that while traditional approaches using pretrained transformers requires fine-tuning them for the desired task, it is **NOT** required to fine-tune the transformers to pass the hard baseline. Using frozen weights are sufficient to pass the hard baseline. To do so, one can run the transformer separately and save the results to a stand alone file. When needed, the results are loaded without additional computation. It is recommended to refer to previous tasks for examples on how this is done. Interested students with sufficient computational resources are encouraged to fine-tune the models. This best mimics real world use cases of pretrained transformers.
Due to the heavy computational cost associated with transformer inference, it is recommended to use a modern GPU for this task. Students are invited to use Google Colab or the ETH Euler cluster. If no GPU can be obtained, it is possible to pass the hard baseline using CPU-based computation on the ETH Euler cluster within a reasonable time limit.

EVALUATION

We are interested in accurate prediction of the review score from the content. Your submitted predictinos will be evaluated by the *mean square error (MSE)* to the true score. Lower is better. The MSE is defined as

$$\mathrm{MSE}(y_*, y) = 1 N N \sum_{i=1} (y_{*i} - y_i) 2$$

SUGGESTED MATERIAL (OPTIONAL)

You will learn all the details about transformers in the last few lectures of this course. However, if you are interested about the inner workings of transformer models the following will prove useful.

- Last year's IML material on transformers.
    - Lecture slides.
    - Lecture recording
    - Tutorial recordings
- Hugging face tutorials on transformers.
- 3Blue1Brown. But what is a GPT? Visual intro to transformers.
- Attention is all you need.