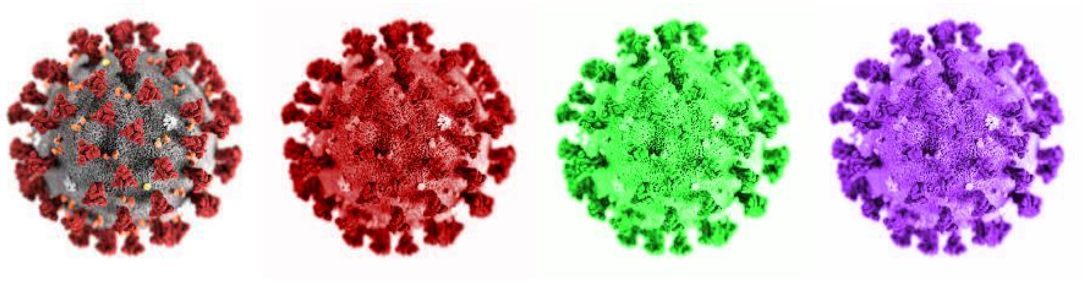
HW3: Regression

In this assignment, we will try to predict the presence of a new COVID variant.

Scientists have developed an advanced test that computes a probability that a patient is infected with a new COVID variant. Unfortunately, this test is costly and cannot be performed in large scales.

Our goal is to regress the outcome of this test, i.e., the "variant score", with the cheap features we used in the previous assignments.

Good Luck!



Instructions

- Submit in pairs by **Thursday, 01.07.2021**, 23:59.
  - Late submissions will be penalized 5 points per day.

- **Your code**
  - Should be clearly and briefly documented.
  - Variables/classes/functions should have meaningful names.

- **Final report**
  - **Should not contain the code itself.**
  - **Do not submit jupyter notebooks as PDFs.**
  - Should be written in a word processor (Office Word, Google docs, etc.).
  - Can be in Hebrew, English, or both.
  - Answer the questions in this instruction file according to their numbering.
  - Add concise explanations, figures (outputs of your code), tables, etc.
  - You are evaluated for your answers but also for clarity and aesthetics.
  - Plots should have suitable titles, axis labels, grid lines and legends (if needed).

- **Grading**
  - 20% of the grade is based on your model accuracies and your code:
    - In Section 8 you submit predicted labels for unlabeled data.
    - 10 points are given to all submissions achieving MSEs lower than some thresholds which we do not publish (should be easy to achieve).
    - 10 points are based on a competition between teams, see Section 7.
  - The other 80% is based on your final report. Note that some answers are dependent on your personal results. Not all questions have a single answer.

- **Submit a zip file containing** (please use hyphens, not underscores):
  - The report PDF file with all your answers, named `id1-id2.pdf`.
  - Your code (choose the relevant options for you):
    - Working with jupyter: a notebook with your code, named `id1-id2.ipynb`.
    - Working with a "traditional" IDE: one clear main script, named `id1-id2.py`, and any additional files required for running the main script.
  - Your predictions files, named `pred_{3,4,5,7}.csv` (see Section 8).

Section 1: Quick data exploration and preparation

Load the labeled data from *variant_labeled.csv*.

Notice:

- We deleted some uninformative features (e.g., `Address` and `PCR_15`).
- Remaining features come from the same distributions as in previous assignments.
  - The correlations between remaining features are unchanged.
- We now have a new target variable: `VariantScore`.
  - The correlations between the features and the target variable are unknown. That is what we need to investigate!
  - Not all remaining features are necessarily useful.

Splitting the data

(Q1) Split the data into a train set (80%) and a test set (20%). Like in HW1, set the `random_state` to be the sum of the last digits of both your ID numbers. Notice: do not create a separate validation set, since we will use k-fold validation.

Before preprocessing

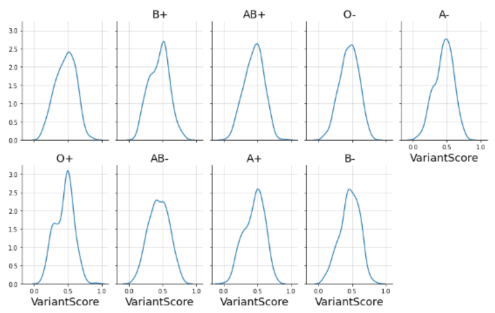
The following steps should be done on the train set only.

(Q2) Use kdeplot to plot the conditional distribution of the target (`VariantScore`) given the `Sex` of the patient. Can you identify a pattern?

Notice: in this assignment the `Sex` feature has no missing values (`NaN`).

(Q3) Use FacetGrid to plot the kdeplot conditional distributions of the target (`VariantScore`) given the `BloodType` of the patient.

The result should roughly look like the illustration to the right (the distributions here are made up).



You can use the following code snippet:

```
g = sns.FacetGrid(S_train, col="BloodType", height=3.5, aspect=.65, col_wrap=5)
g.map(sns.kdeplot, "VariantScore")
g.set_titles(col_template="{col_name}", size=18)
g.set_xlabels(size=18)
for ax in g.axes:
    ax.grid(alpha=0.5)
```

Can you identify a pattern?

Preprocessing (data preparation)

All preprocessing decisions (imputation, outliers, cleaning, normalization, etc.) should be made according to the train set only. After deciding on the suitable preprocessing operations, apply them to both the train and test sets.

The output of the preparation step should be the two datasets in a tabular format that is suitable for linear regression and other regression algorithms. That is, all features should be numeric (no strings, no missing values) and preferably normalized.

You are free to use whatever preprocessing steps you wish, according to:

i. Your insights from the previous assignments.
ii. Further data exploration steps you use now.
iii. Insights from following sections (but be **careful** not to rely on the test performance!).

Remember, the better the features – the better is the performance of your models.

However, you cannot add additional features to the dataset, until you reach the custom models in Section 7.

Finally, answer the following questions:

(Q4) Based on Before preprocessing, explain how you converted the `Sex` feature into a numeric feature (or features). Briefly explain your reasoning.

(Q5) Based on Before preprocessing, explain how you converted the `BloodType` feature into a numeric feature (or features). Briefly explain your reasoning.

(Q6) Briefly explain which other preprocessing steps you performed (imputation, cleaning, normalization, outliers, etc.).

Your answer should take between half a page and a full page.

You can also answer this question (here) after you finish sections 3, 4, and 5.

Section 2: Evaluation

Throughout this assignment, we focus only on regressing the `VariantScore`.

Our general goal is to minimize the generalization MSE, that is $\min_{h} \mathbb{E}_{(x,y)\sim\mathcal{D}}[(h(x) - y)^2]$.

As we have learned, in practice, we instead minimize the empirical error $\frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$ on a train set, and tune hyperparameters using a validation set.

In this assignment we will use the k-fold cross-validation method for better estimating the generalization error, thus improving the tuning procedure. We will use $k = 5$ folds.

Complete the following method[1] that computes the mean squared error (train and validation) of a given untrained regressor `h` on a given train set using `n_splits` folds.

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import mean_squared_error, make_scorer

def CV_evaluation(h, X_train, y_train, n_splits=5):
     scores = cross_validate(h, X_train, y_train, cv=n_splits,
                        scoring=make_scorer(mean_squared_error),
                        return_train_score=True)

     # TODO
     train_mse = 0
     valid_mse = 0

     return (train_mse, valid_mse)
```

Let us test your method on a naïve baseline regressor to which we will compare our regressors throughout the assignment.

(Q7) Create a DummyRegressor that always predicts the average `VariantScore` of the train set. Evaluate its performance using `CV_evaluation` and fill the table below:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2		

Retrain the dummy regressor on the entire train set and save it for future use (Section 6).

[1] In practice, researchers use the GridSearchCV for tuning parameters using cross validation, but here we have reasons to use custom methods.

Section 3: Basic linear regression

Here and in the following sections we will tune hyperparameters. Specifically, in linear regression with regularization, we need to tune the regularization strength (λ in our notation, `alpha` in sklearn).

The repeated tuning process (for a single parameter) should include:

i. Determining the tested values of the tuned hyperparameter (see numpy.logspace). You need to choose suitable values by yourself that will help you optimize the validation error.
ii. For each value, evaluating a suitable regressor using `CV_evaluation`.
iii. Plotting the train and validation errors as a function of the hyperparameter. Consider using semilogx or loglog plots. Also plot a constant line with the validation error of the baseline dummy regressor.
iv. Reporting the value that yields the optimal validation error and its respective error.

We will now learn to predict `VariantScore` using simple linear regressors.
Make sure your models are non-homogeneous (`fit_intercept=True`).

(Q8) Tune the regularization strength of a Ridge regressor.
Remember to attach the required plot and the optimal strength with its validation error.

(Q9) Fill in the train and validation errors of the regressor yielded by the best performing hyperparameter.
Remember to compute these errors using `CV_evaluation`.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	filled	filled
Basic linear	3		

Using the best performing hyperparameter, **retrain** a regressor on the entire train set and:

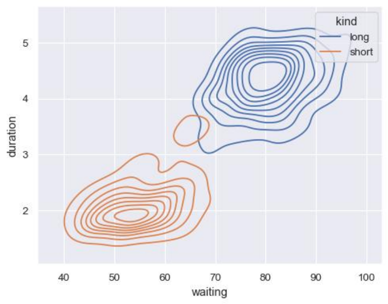
(Q10) Specify the 5 features that have the 5 largest coefficients (in absolute value) in the resulting regressor, from the largest to the smallest (among these 5).

Save this basic regressor for future use (Section 6).

Section 4: Hierarchical linear regression

During the data exploration section, we revealed that the `VariantScore` has very different conditional distributions depending on the `Sex` of the patient. Let us further examine the effect of this feature on the target variable.

(Q11) Use kdeplot to plot the bivariate distribution of `VariantScore` and `AgeGroup`, conditioned on the `Sex` of the patient (like illustrated on the right).
Explain why this behavior cannot be explained by a linear model trained on the raw features (like in the previous section).



We will now examine a simple hierarchical model (i.e., multilevel model) which learns one linear regressor for males, and another one for females.
Start by splitting the training set into two subsets – female patients and male patients.
Think: do you still need the `Sex` feature in these subsets?

(Q12) Tune the regularization strength of a Ridge regressor for each subset separately.
Remember: attach required plots, optimal strengths, optimal validation errors.

We will now construct a multilevel model. Complete the following class to create a custom sklearn regressor that trains a different regressor for each sex.

Fill in the `TODO`s or edit this class as you wish.

```
from sklearn.base import BaseEstimator, RegressorMixin
class MultiRegressor(BaseEstimator, RegressorMixin):
    def __init__(self, h_male, h_female):
        self.h_male = h_male
        self.h_female = h_female

    def fit(self, X, y):
        self.h_male.fit(TODO)
        self.h_female.fit(TODO)
        return self

    def predict(self, X):
         # X should be a pandas dataframe
        all_predictions = []

        for index, x in X.iterrows():
            y_pred = TODO
            all_predictions.append(y_pred)

        return all_predictions
```

For each subset (females and males), find the best regressor you trained in this section, and use it to create a combined multilevel regressor.

For instance,

```
    h_combined = MultiRegressor(Ridge(alpha=2), Ridge(alpha=.1))
```

(Q13) Fill in the additional line in the table using the multilevel regressor (following the previous questions where you tuned two separate regularization strengths).

Remember to compute the errors using `CV_evaluation`.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	filled	filled
Basic linear	3	filled	filled
Multilevel linear	4		

(Q14) Briefly discuss why you got a higher/lower train error and a higher/lower validation error comparing to the basic linear regressor from Section 3.

Train the multilevel regressor on the entire train set and save it for future use (Section 6).

Section 5: Polynomial fitting

We will now try to improve our linear models by adding quadratic features.

Create copies of your train and test sets, and add quadratic features without interaction features. For instance, an example $x = [x_1, x_2]^\top$ should be mapped to $\phi(x) = [x_1, x_2, x_1^2, x_2^2]^\top$ without $x_1 x_2$. Notice: not all features should be added, e.g., if $x_j \in \{0,1\}$ then $x_j^2 = x_j$.

We will now run our regression models using the described polynomial feature mapping.

(Q15) Explain how we can expect the training error to change when using such a mapping.

(Q16) Explain how we can expect the validation error to change when using such a mapping.

Repeat the multilevel procedure from Section 4 by fitting separate models for each sex.

(Q17) Tune the regularization strength of a Ridge regressor for each subset separately. Remember: attach required plots, optimal strengths, optimal validation errors.

(Q18) Using cross validation on the separate subsets, evaluate the tuned regressors from this section and those from the previous Section 4. Fill in the following table:

Multilevel Model	Section	Sex	Train MSE	Valid MSE
			Cross validated	
Linear	4	**M**		
Polynomial	5			
Linear	4	**F**		
Polynomial	5			

(Q19) For each sex, briefly discuss whether the feature mapping helped to model the `VariantScore` of patients of that sex from an overfitting and underfitting perspective.

Using the best regressors you tuned in this section, create a multilevel regressor (after the feature mapping, using the custom `MultiRegressor` class).

(Q20) Fill in the additional line in the table using the combined regressor (after the previous questions where you tuned two separate regularization strengths).

Remember to compute the errors using `CV_evaluation`.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	filled	filled
Basic linear	3	filled	filled
Multilevel linear	4	filled	filled
Multilevel poly.	5		

Train the multilevel regressor on the entire train set and save it for future use (Section 6).

Section 6: Testing your models

Important: do not continue to this section until you have finished all previous sections.

Finally, we can let the test set come out and play.
At the end of sections 2, 3, 4, and 5, you retrained the tuned models on the entire train set.
Now evaluate the test errors (MSE) for these 4 models.
Remember: do not cross-validate here. Train on the train set and evaluate on the test set.

(Q21) Complete the entire table.

Model	Section	Train MSE	Valid MSE	Test MSE
		Cross validated		Retrained
Dummy	2	filled	filled	
Basic linear	3	filled	filled	
Multilevel linear	4	filled	filled	
Multilevel poly.	5	filled	filled	

(Q22) Which model performed best on the test set?

Discuss the results in the table (from an overfitting and underfitting perspective, or any other insightful perspective).

Section 7: Custom models challenge (mandatory)

This is the section where you are given the freedom to choose your own models. Train a regression model (any model) from the sklearn library only. You can train multilevel models (not only for different sexes), ensembles of models, etc. Other directions include Lasso, ElasticNet (combines L1 and L2 regularization), gradient boosting trees, multilayer perceptrons (neural networks), and more.

You should remember that better features will give you better results. You can use any feature mapping you like and/or employ any insight you have on the raw features and their interactions with each other or with the target variable.

Your goal

Find a model that achieves the lowest generalization MSE (think how to estimate it in the best way). You should train using the labeled dataset you worked with so far, and finally predict labels for the unlabeled dataset in the next section.

Competition and Grading (10pts, not a bonus)

In the next section, you will predict labels for an unlabeled dataset and submit them. We will compute the MSE of your predictions using the ground truth labels.

Finally, we will rank the submissions according to their MSEs and grade this competition uniformly. That is, the submission with the lowest MSE will receive 10pts, and the one with the highest MSE will receive 0pts.

Clarification: if you are short on time, you can simply use a model from a previous section, but this will probably result in a low grade for this competition.

(Q23) Explain which model you used in this section. Describe any special feature mappings you used or any additional data transformations you used. Describe the selection process you used. This question should take between half a page and a full page.

Section 8: Submitted model predictions

For each model you implemented, you should submit a single CSV file containing your predictions on the unlabeled dataset `variant_unlabeled.csv`.

Notice that this unseen dataset requires the same preparation as the rest of your data.

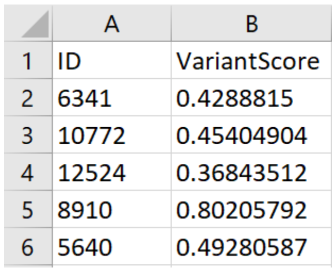
Before performing any predictions, train each of your models on the **entire labeled dataset at your disposal**, including the labeled test set.

For each relevant section, submit a file named `pred_<sect_num>.csv`.

Each file should consist of an “`ID`” column and a “`VariantScore`” column.

Every row should have the original ID of the predicted input and the prediction of the best regressor you found in the appropariate section.

Example for a prediction file (actual file should have 2001 rows including the headers):



	A	B
1	ID	VariantScore
2	6341	0.4288815
3	10772	0.45404904
4	12524	0.36843512
5	8910	0.80205792
6	5640	0.49280587

Submit the following files (all mandatory):

- Non-comparative sections (10pts):

  Here we simply verify that your predictions “make sense” and MSEs are low enough.

  - **`pred_3.csv`** Best basic linear model’s predictions.
  - **`pred_4.csv`** Best multilevel linear model’s predictions.
  - **`pred_5.csv`** Best multilevel polynomial model model’s predictions.

- Comparative section (10pts):

  Here you compete with the other submissions, see previous section.

  - **`pred_7.csv`** Best custom models’ predictions.

NOTE: do not submit predictions with a false model, e.g., `pred_3.csv` with multilevel predictions. We know the limitations of the models.