Exercise 3 – The Virus Challenge –Modeling

Winter is coming, and the national health authority asks for your assistance in the several classification tasks. For every patient:

- Detect virus type and presence
- Decide whether he is "at risk" patient
- Decide whether he is a potentially "super-spreader"
- Identify which factors (features) are more significant for the above classification tasks

Mandatory Assignment

Wet (80%)

First, do the data preparation task again (of the former exercise), focusing on the right set of features (see the first comment below). Specifically, write a script that implements the following:

1. Load the Virus Challenge data from the *virus_hw2.csv* file
   - The data is the same as in the 2nd exercise
2. Select the right set of features, and apply the data preparation tasks that you have carried out in the former exercise, on the train, validation, and test data sets
   - At the very least, fill up missing values
     - All other actions, such as outlier detection and normalization, are not mandatory. Still, you are encouraged to do them, as they should provide you added values for the modeling part
   - Whether to use the validation set for pre-processing steps is your call
3. Save the 3x2 data sets in CSV files

Next, write a Python script that can handle the following prediction tasks

- Detect virus type and presence
- Decide whether the patient is "at risk"
- Decide whether the patient is a potentially "super-spreader"

Such a process should implement and execute the following

1. Load the preprocessed training set
2. Train at least two models
   - Each training should be done via cross-validation on the training set, to maximize performance of the model while avoiding overfitting
3. Load the prepared validation set
4. Apply the trained models on the validation set and check performance
   - It is your call which performance measure to use, and it is possible to check multiple measures
5. Select the best model for the prediction tasks
   - The model selection can be "manual" (not an automatic process), but it should be based on the performance measurements

6. Use the test set to evaluate your model
    - For every patient in the test set:
        i. Detect virus presence
        ii. Decide whether he is classified as "at risk" patient
        iii. Decide whether he is classified as a potential "super spreader"
    - Provide appropriate model metrics for every of the three classification tasks
7. Predict the classes for the new unlabeled data from the "virus_hw3_unlabeled.csv" file. Output your results to another csv file named "predicted.csv".
    - This file should contain 2 columns only ("PatientID", "TestResultsCode"). You can use the "predicted.csv" file from the HW3 section in the webcourse as a reference.

Please submit

1. The Python script file that implements the data preparation part using the right set of features
2. CSV files of the prepared train, validation, and test data sets
3. The Python script file that implements the modeling (training and model evaluation) part
4. A CSV file that contains the class predictions (predicted labels) for the unlabeled data from "virus_hw3_unlabeled.csv" file.
    - Use the file "predicted.csv" as a reference
5. A short documentation that
    - Explains your process and any significant decisions/insights you would like to share. Specifically, explain the following:
        i. A concise description of your (updated?) process of data preparation
        ii. Your choice of models and hyperparameters
        iii. Your model evaluation strategy
    - Includes answers for the following
        i. What features are more significant for each classification task (virus type/patients at risk/ potential super spreaders)? Explain how you identify these key factors.
           Notes:
            1. Provide a list that maps each of the 3 classification tasks to their corresponding significant features.
            2. Handle this task strictly from a technical perspective, meaning please ignore the semantic of the features
    - **The answers to the dry part.**

Dry (20%)

1. Consider the least squares problem with a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ and a vector $\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{m}}$:

$$\text{argmin}_{\boldsymbol{w}} \|\mathbf{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2.$$

Denote $r = \text{rank}(\mathbf{X}) \leq \min\{m, d\}$.

Denote the <u>full</u> SVD of the data matrix $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{\min\{m,d\}} \sigma_{i,i} \boldsymbol{u}_i \boldsymbol{v}_i^\top$ where $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times d}$ and $\mathbf{U}, \mathbf{V}$ are square orthonormal matrices of appropriate sizes.

Prove that $\widehat{\boldsymbol{w}} = \mathbf{V}(\boldsymbol{\Sigma}^+)^2 \boldsymbol{\Sigma}^\top \mathbf{U}^\top \boldsymbol{y}$ is an optimal solution to the least squares problem, where for the singular values matrix $\boldsymbol{\Sigma}$ we get $(\boldsymbol{\Sigma}^+)_{i,j} = \begin{cases} 1/\Sigma_{i,j}, & i = j \leq r \\ 0, & \text{otherwise} \end{cases}$.

2. We wish to better understand the SVM objective.

Recall that a valid kernel function $K(x, x')$ must hold $K(x, x') = \langle \phi(x), \phi(x') \rangle$ for some feature map $\phi: \mathbb{R}^d \to \mathbb{R}^D$. A necessary and sufficient condition is that K is positive semidefinite. That is, for any set of points $x_1, \ldots, x_m \in \mathbb{R}^d$, the Gram matrix $\mathbf{G} \in \mathbb{R}^{m \times m}$ defined by $\mathbf{G}_{i,j} = K(x_i, x_j)$ is positive semidefinite. Recall that a symmetric $\mathbf{G}$ is PSD if and only if $\forall \boldsymbol{z} \neq 0_m: \boldsymbol{z}^\top \mathbf{G} \boldsymbol{z} \geq 0$ (or equivalently, all its eigenvalues are non-negative).

Recall the SVM optimization problem from lecture 05 which uses the Gram matrix:

$$\min_{\boldsymbol{\alpha}} \underbrace{\overbrace{\lambda}^{\geq 0} \boldsymbol{\alpha}^\top \mathbf{G} \boldsymbol{\alpha} + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i (\mathbf{G}\boldsymbol{\alpha})_i\}}_{= \mathcal{L}_{SVM}(\boldsymbol{\alpha})}$$

2.1. Consider two valid kernels $K_i(x, x') = \langle \phi_i(x), \phi_i(x') \rangle$ for $i = 1,2$.

Show that the following functions are also valid kernels by explicitly writing their respective feature maps ϕ_3, ϕ_4 in terms of ϕ_1, ϕ_2.

2.1.1. $K_3(x, x') = K_1(x, x') + K_2(x, x')$.

2.1.2. $K_4(x, x') = f(x) \cdot f(x') \cdot K_1(x, x')$ for any function $f: \mathbb{R}^d \to \mathbb{R}$.

2.2. We will now understand what can go wrong when $\mathbf{G} \nsucceq \mathbf{0}$. Consider a trainset with two points $\{(x_i, y_i)\}_{i=1,2}$, a "kernel" holding $K(x_1, x_1) = K(x_2, x_2) = 1$ and $K(x_1, x_2) = 2$, and labels $y_1 = 1, y_2 = -1$.

2.2.1. Write down the Gram matrix, its two eigenvectors and its two eigenvalues. You can use [numpy.linalg.eigh]() to compute them numerically.

2.2.2. Find a vector $\boldsymbol{v} \in \mathbb{R}^m$ such that $\lim_{c \to \infty} \mathcal{L}_{SVM}(c \cdot \boldsymbol{v}) = -\infty$.

Prove that the vector you suggested holds the above limit.

<u>Note</u>: understand why this demonstrates that non-PSD "kernels" are problematic.

2.3. We will now show that this cannot happen for any $\mathbf{G} \succcurlyeq \mathbf{0}$.

2.3.1. Prove that $\min_{\boldsymbol{\alpha}} \mathcal{L}_{SVM}(\boldsymbol{\alpha}) \geq 0$.

2.3.2. Prove that $\min_{\boldsymbol{\alpha}} \mathcal{L}_{SVM}(\boldsymbol{\alpha}) \leq 1$.

Non-Mandatory Assignments (10%)

The following list includes additional, non-mandatory, assignments. You are highly encouraged to do at least some of them. Each functional implementation of ANY assignment will get a bonus

A. Automate the model selection procedure, i.e. the selection of the best model based on the performance measurements of all the trained models (Step 5 of the mandatory process)
   - Provide a Python script file and a document that explains the process, your insights, and conclusions.

B. It may very well be that "one size doesn't fit all", namely that different models may bring better results in different tasks Check this paradigm -
   - Use a different modeling procedure (train and test) for each of the three mandatory prediction tasks
   - Compare results with the results obtained using the one model approach
     - Note that "Better results" are not merely a higher accuracy, but also simpler models (why is it important?), stable predictions, etc.
   - Provide a Python script file for each of the tasks and a document that explains the process, your insights, and conclusions.

Comments

- **The right feature set –**
  Since there is a redundancy in the original set of features, the "right" set of features is not unique (meaning, there are a few subsets of features that could have been selected). We may publish a mandatory feature set in the future.

- You may use a "balanced" training set (to handle imbalanced classes while training a model) but the test set should reflect the original data distribution, so that the reported performance measurements will be unbiased