Practical task 4.

Optimization: GD vs SGD Consider the logistic regression method for binary classification. In this problem you need to implement gradient descent (GD) and stochastic gradient descent (SGD) methods for log-loss optimization.

1. Consider the log-loss with L1 regularization: $L(w) = \sum_i ln(1 + exp(-w^T x_i y_i)) + \gamma \|w\|_1$. Find its derivative and the update rules for GD and SGD.

2. Implement `Python` functions that optimize the log-loss with GD and SGD. Prototypes for these functions are the following:

   ```
   w,L = GD(X_train, y_train, max_epoch = 1000, alpha = 0.1, gamma = 0, tol = 0.1)
   w,L = SGD(X_train, y_train, max_epoch = 1000, alpha = 0.1, gamma = 0)
   ```

   where `w` is a weight vector, `L` is a vector of loss values after each epoch, `alpha` is a step size, `tol` is a tolerance parameter for GD stopping criterion and `gamma` is a coefficient for regularization.

   Some hints for you:

   - Weights initialization: zeros or small random numbers from $[-\frac{1}{2d}, \frac{1}{2d}]$.
   - Use $\nabla_w L/N$ instead of $\nabla_w L$ for GD, where N is a number of objects. In this case efficient step size is approximately $0.01 - 1$.
   - Step size: constant for GD and decreasing for SGD, for example, $\alpha/epoch\_number$ where α is some constant.
   - Random order of objects for SGD. Use `sklearn.utils.shuffle`
   - Stopping criteria: for GD use $L_{old} - L_{new} < tol$, for SGD simply do a particular number of iterations.
   - Regularization: use L1 norm of weights vector as a regularizer.

3. Generate simple 2-dimensional data and check that your functions work properly on it. Don't forget to standardize the data and then add a constant feature to it. Run GD and SGD from the same initial point and compare the results. What can you say about the convergence rate of these two algorithms?

   Here you should demonstrate the following plots:

   - plot with data points and decision boundary for GD and SGD,
   - plot of decreasing L for increasing epoch number (for GD and SGD together).

Regularization

1. Load the first dataset. Use `pickle.load`. Fit a logistic regression classifier on the training samples. Use GD with and without regularization, start both methods from the same initial point (for example, use zero vector as initial weights vector). Don't forget to standardize the data and then add a constant feature to it.

2. Compare the results of the methods on the train and test data, explain the difference.

3. Use the resulting weights vector of GD with regularization to determine two the most important features. Fit the logistic classifier only on these two features (+ the constant one) and visualize the decision boundary. Does L1 regularization help you to chose important features?

Model evaluation In this part of the task you will work with the problem of diabetes diagnostics. Load the diabetes dataset using `pickle.load`. This dataset has the following features:

1. Number of pregnancies
2. Plasma glucose concentration after 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure
4. Triceps skin fold thickness
5. 2-Hour serum insulin
6. Body mass index
7. Diabetes pedigree function
8. Age

Class label is equal to 1 if a person has a diabetes and to -1 otherwise.

1. Train the logistic regression classifier on this dataset. Use SGD without regularization. Don't forget to standardize the data and then add a constant feature to it.
2. In diagnostic problems false positive and false negative errors actually have different costs. Let's say, if we make false negative error (don't detect a condition when it is present), then the patient doesn't have a necessary treatment and, if we make false positive error (detect a condition when it isn't present), then the patient simply need to be tested more. Therefore, the cost of false negative error is higher and we care much more about this type of error.

   Compute a confusion matrix for fitted classifier. How many errors of each type have you got? Compute a false positive and a false negative rates for this classifier. Why are they so different?

   Useful functions: `sklearn.metrics.confusion_matrix`.
3. To change the proportion of errors of different types you can change a threshold a at the prediction rule $y = \sigma(w^T x) > a$, where $a \in [0, 1]$.

   Show the ROC-curve of the fitted classifier and a point on it which corresponds to $a = 0.5$ (the one you computed at the previous step). Using ROC-curve choose a so that false negative rate is less than 20% while false positive rate is still small. What accuracy and false positive rate does the final algorithm have?

   Useful functions: `sklearn.metrics.roc_curve`.