Practical task 2.

Curse of dimensionality

1. For each dimensionality $d \in [1, 200]$ generate a dataset of 100 random points from uniform distribution on $[-1, 1]^d$.

   Useful functions: *numpy.random.uniform*.

2. Write the function that calculates Euclidean distances from the given point x to its nearest and farthest neighbors in the dataset (minimum and maximum distances).

   Useful functions: *scipy.spatial.distance.cdist*, *numpy.amax*, *numpy.amin*.

3. Plot average minimum and maximum distances for varying d (here you should calculate average values on dataset). Also plot their ratio — average maximum divided by average minimum.

Does the curse of dimentionality take place here? Explain your answer.

Visualization of the decision surface

1. Generate two synthetic datasets from two-dimensional normal distributions with $\mu_1 = (3, 3)$, $\mu_2 = (1, 1)$ and identity covariance matrices. Visualize these datasets.

   Useful functions: *numpy.random.multivariate_normal*, *pylab.scatter*.

2. Assign label 0 to the items from the first dataset and label 1 to the items from the second dataset. Join these datasets into one and fit kNN and decision tree on the resulting dataset (with default hyperparameters).

   Useful functions: *sklearn.neighbors.KNeighborsClassifier*, *sklearn.tree.DecisionTreeClassifier*.

3. Visualize the obtained decision surface for each algorithm. Why do these surfaces look like this?

   Useful functions: *numpy.meshgrid*, *pylab.pcolormesh*.

kNN and decision tree comparison In this part of the task you should predict whether income of a person exceeds $50K$.

1. Load the data from csv-file.

2. Prepare the data for a classification algorithm. Construct the vector of labels y and two different feature matrices:

   - matrix X_1 that contains only numerical features (you should drop categorical features),
   - matrix X_2 that contains all features (you should use one-hot encoding for categorical features).

   Useful functions: *pandas.DataFrame.drop*, *pandas.get_dummies*.

3. Split the data into random train and test subsets in proportion 70:30. You should split X_1, X_2 and y together.

   Useful functions: *sklearn.cross_validation.train_test_split*.

4. Compute standardized versions of feature matrices X_1 and X_2. Here you should fit a StandardScaler only on a train subset and then transform both train and test subsets.

   Useful functions: *sklearn.preprocessing.StandardScaler*.

5. Fit kNN-classifier and decision tree to four train datasets: with/without categorical features and with/without standardization. For each algorithm on each dataset find the best parameters: k for kNN-classifier, maximum depth of a tree (*max_ depth*) and minimum number of samples required to be at a leaf node (*min_ samples_ leaf*) for decision tree. Use accuracy on the test subset (the fraction of correctly predicted labels) as a quality measure.

   Useful functions: *sklearn.neighbors.KNeighborsClassifier*, *sklearn.neighbors.DecisionTreeClassifier*, for both classifiers - *fit*, *predict*, *score*.

6. Compare all fitted classifiers:

   - Which classifier is the best for the given problem?
   - How does standardization influence on the accuracy of kNN-classifier? And what about a decision tree? Explain these effects.
   - Which classifier works better with categorical features? Explain why using categorical features without standardization isn't a good choice for kNN-classifier.

Bonus(extra points)

* Visualize the decision tree fitted to dataset without categorical features and without standardization (with the best parameters). Which features are the most important in this task based on the tree structure?

Useful functions: *sklearn.tree.export_ graphviz*, *pydot.graph_ from_ dot_ data*.

Add a picture of the tree to the email with the task.

* Choose two most important features based on the tree structure and visualize a decision boundary of this tree in the corresponding 2-dimensional space.