Slide 1

Good afternoon and welcome to the fifth lecture of the Machine Learning course. Today we’ll talk about how to work with attributes for models and how to choose models from many existing models. Today's lecture is not directly related to machine learning, because the goal of machine learning is to search for patterns in data using mathematical models. Assessing models and choosing from existing models is the next step after machine learning and it is more tied to the problem you are solving. We will also talk about preprocessing features in data. This topic is even more dependent on your task.

Slide 2

So, here is the plan for today's lesson. We will start with a topic that each of you has already touched on in laboratory work, namely we will talk about methods for evaluating machine learning models. We will touch on the most basic methods for calculating the quality of a machine learning model, both when solving the classification and regression problems. We will talk about the simplest assessment methods since they form the basis for more complex methods and these methods are not tied to the specific problem being solved. Then we will talk about how to use the obtained estimates to compare models among themselves. Of course, there are a large number of criteria for comparing models. They can be, for example, the complexity of the calculations, the time required for training, the ability to retrain the model. But we will talk about methods for choosing models based on their quality, i.e. based on how many useful patterns the machine learning algorithm was able to detect. Then we summarize everything we talked about in the so-called machine learning pipeline. This is a top-level description of the process of working with data in machine learning. Finally, we talk about extracting features from raw data and preprocessing them before using machine learning algorithms.

Slide 3

So, let's start talking about methods for evaluating machine learning models. This question may seem strange, but we must ask this question in order to better understand the topic. Why do we even need to measure the quality of the machine learning model? Such a simple and naive question actually contains a lot of answers. First of all, we need to understand how well the machine learning algorithm has identified patterns in the data. For the model must be measurable. Measurement of model quality is one of the ways to configure machine learning model parameters and compare models among themselves, and we will talk about this later. Moreover, when you start using the machine learning model in production, you will need to somehow understand how the algorithm responds to new data that might not have been in the training set. In all of this, methods for evaluating machine learning algorithms will help us.

Slide 4

And we will start with the basic terminology for binary classification. Let me remind you that in binary classification we are faced with the task of correlating the objects of the training sample with one of two classes. This could be, for example, a spam filtering task. There are two classes in this task - spam and not spam. Based on the fact that two classes are present in the problem and, accordingly, the model should be a function that returns one of two values, there can be four combinations of the correct answer and the model answer. True-Positive, False-Positive, True-Negative, and False-Negative. Imagine that in this problem there are two classes 0 and 1. The value 0 corresponds to the Negative class, and the value 1 corresponds to the Positive class. In this case, the model may be mistaken and return the opposite answer, or it may return the correct answer. Thus, True-Positive means that the correct answer was 1 and the model was not mistaken. False-Positive means that the correct answer was Negative, but the model responded with Positive. Similarly, for label 0. TP, TN, FP, and FN are the error and correct answer counters in the binary classification, which are used as the basis for calculating model quality metrics.

Slide 5

On this slide you can see how these counters are used to evaluate the quality of the binary classification. For example, accuracy shows the share of erroneous classifications among all Positive cases in a training or test sample. At the same time, Recall reflects how many Positive responses in total the machine learning algorithm was able to determine among the known Positive answers in the test set. Accuracy shows the total error rate for the test set. And the F-score is a metric that combines precision and recall, since it is often important to evaluate both and compare models against each other based on these metrics.

Slide 6

Before describing the following metrics, it is necessary to review the concept of threshold for a model. Most classification models can be adapted to assess the degree of confidence in a model’s response or to assess the likelihood of a response. Then, if such an assessment exists, we can choose a certain threshold value after which we will assume that the model is sufficiently confident in its answer. For example, if the model estimates the probability that the message is spam at 60 percent, then with threshold equal to 0.5 we will consider the message to be spam, but at threshold equal to 0.7 we will not consider the message to be spam and consider it to be a regular message. One of the ways to evaluate the model consists precisely in assessing how strongly the model is sensitive to its threshold values. Let's move on to methods for evaluating models that use the concept of a threshold.

Slide 7

The first evaluation method is a Precision-Recall curve. This is a graph showing how changing the threshold value of a model changes the score of Precision and Recall of the model. The essence of this method is to sequentially sort through the threshold value with a certain step, set it for the model and evaluate the characteristics of Precision and Recall taking into account the set threshold. In the pseudo-code on the slide, you can see how TP, FP, and FN counters are explicitly counted for the model and how to use them when calculating Precision and Recall metrics. Going through the threshold values ​​from 0 to 1, we can build a Precision-Recall graph.

Slide 8

By the way, the scikit learn library already has an implementation of the algorithm for constructing this graph. So you do not need to do your own implementation of this method. An example of use is shown on this slide.

Slide 9

Finally, here is an example of how this chart might look. What can it be useful for? Firstly, you can see from it what precision and recall values ​​you can generally expect from a model. Secondly, and this is the most important part of the schedule, we can make precision and recall choices for our task. In some tasks, precision will be the most important characteristic. These tasks include medical diagnosis and treatment choice. We cannot begin to treat a person if the precision of the algorithm is below 0.99, because the lives of other people depend on this. On the other hand, recall may be important in some tasks, when it is not so scary to make a mistake in classification than to skip a positive-class object. An example of such a task is the automatic rejection of low-cost products in production. Thirdly, the graph shows the overall quality of the model and allows you to compare with other models. The larger the area under such a graph, the greater the precision algorithm for large recall values. This is a very important characteristic, showing the stability of the algorithm.

Slide 10

Another idea for evaluating a classification algorithm is the ROC-AUC or area under curve of receiver operating characteristic. This indicator is equal to the area under the curve, which shows the ratio of the correct classifications to the erroneous ones when the threshold value of the classifier changes. In general, the calculation algorithm has not changed much except that now we not only plot this curve, but also estimate its area.

Slide 11

Here is an example of a ROC chart. The false positive rate is located on the X-axis of this graph - this is the frequency of erroneous classifications, and the true positive rate is located on the Y-axis. These are two more characteristics for evaluating the binary classification algorithm. You can see their calculation in the pseudo-code on the previous slide. The essence of this measure is approximately the same as that of precision-recall graphics - the larger the area under the graphic, the better. A large sweat area in such a graph means that the algorithm is insensitive to threshold changes and shows good quality over the entire range of threshold values. Pay attention to the gray line. If you get a graph that coincides with such a line for binary classification, then this means that the algorithm works no better than tossing a symmetric coin.

Slide 12

In the scikit learn library you can find algorithms for calculating all these metrics. The slide shows an example of pointwise quality estimates of some classifier. Note that the metrics in scikit learn are already adapted for multiclassification tasks, so you do not have to invent your own metric for a task with more than two classes.

Slide 13

Estimating the regression algorithm in general may seem even simpler than evaluating the classifier. The slide shows the basic quality metrics of regression algorithms. All these metrics are based on calculating the difference between the model answer and the correct answer in the test set. The simplest estimate of MAE or mean absolute error is nothing more than the average of all the errors of the regression model on the set of answers. More interesting is the metric R^2 or the coefficient of determination. This metric shows how much your model is better than simply evaluating the answer as the average of all the answers in the sample. In such a metric, a value of 1 indicates an ideal model that perfectly predicts the correct answers. A value of 0 for the coefficient of determination says that such a model is no better than the average value of the responses in the sample. Moreover, such a metric can take negative values, which will indicate that the algorithm is completely unable to predict the correct answer.

Slide 14

So, we examined several basic methods for assessing the quality of a model. Now we can use them when fetching a machine learning algorithm. Why can we find this useful? Firstly, model comparison methods allow you to configure model parameters during training. We can go through several values ​​of a parameter and choose a model that shows the highest quality. Secondly, the methods of model selection allow you to choose the best machine learning algorithm to solve the problem. Not all algorithms are suitable for all tasks and it may turn out that in your task a simple linear model will show the best quality.

Slide 15

The simplest method for assessing the quality of a model from a test set is the Hold-Out method. He is familiar to all of you. Here is its idea: take a large sample of the data and split it into two parts. A smaller part of this partition will be your test set. You should never look into this data set in order not to accidentally adjust to this data. This leads to overfitting. Most of the set is divided into two more parts. The biggest part that you get is used to learn the machine learning algorithm. The second part is a validation set. This data set is used to analyze the errors of your model. Having trained the model on the largest sample, you evaluate it by the validation set, analyze the model errors and adjust the parameters of the training algorithm. Thus, the validation set is used as a way to select model parameters, and the test set is used as a way to choose among different models. For these two sets, the metrics we just reviewed are calculated and the best model is selected.

Slide 16

In laboratory work, you have already encountered this method. This method is implemented in the train\_test\_split function of the scikit learn library, which divides the set of objects and answers into two sets - the set for training and the set for testing the model. In this case, the parameter of this function is the fraction of the total sample size for testing. Typically, one third of the initial sample is taken for testing. To build a validation set, you can take about ten percent of the original data. The remaining data is used to train the model.

Slide 17

You already got acquainted with the Leave one out method at the lecture about the k-nearest neighbors algorithm. Let me remind you that the main idea of ​​this algorithm is to build a number of models and evaluate them while simultaneously dropping just one element from the training set. It is on this element that the model is evaluated. The pseudo-code on the slide shows this idea. We divide the training set into a series of folds, each of which has only one element for testing the model. Further, the model is trained in turn on each such subset, and if its answer does not coincide with the response of the deferred object, then the error counter is increased. Such error counters are used as model selection algorithm.

Slide 18

Here is an example of using the Leave-one-Out method in the scikit learn library. Please note that this method generates a set of indices by which data is divided. One part of the indices for training and it contains the indices of all the rows from the original training set. The second part of the indexes contains a link to only one row of data - on this row the model will be checked for quality. As a result, the data from the sample is not copied and we can access this data without losing additional RAM.

Slide 19

The second method for evaluating model parameters and comparing them with others is called cross-validation. This method is very similar to LeaveOneOut, but it is a generalization of it. Now, to evaluate the model, not one object is extracted from the initial training set, but a subset, a certain fraction of the initial size of the training set. To evaluate and compare the model with this method, the machine learning algorithm is independently trained on subsets of the test sample and is also independently evaluated by some chosen metric. After the assessment is completed, a series of numerical values ​​is obtained, which corresponds to the calculated model quality metrics. For the resulting number series, averaging is performed and the standard deviation is calculated. The resulting estimate allows not only to select model parameters or compare algorithms with each other, but also it has the property of statistical stability in the sense that such a quality assessment is more correct.

Slide 20

On this slide you can see an example of using cross-validation to evaluate the parameters of a binary classification model. Check out the sklearn.model\_selection package. This scikit learn library package contains a number of tools that allow you to conveniently configure model parameters or select a machine learning algorithm using the considered methods of model selection.

Slide 21

Okay, now we can consider the process of machine learning in the context of the task at hand. Machine learning is just one of the steps in the data mining process. Only in the training task do you get a convenient training set and you do not need to train different algorithms or for a long time to sort through the training parameters. Often in a real task, things are different. The slide shows a machine learning pipeline. Machine learning itself is only part of this scheme, and one of the smallest, since often machine learning refers to a call to an existing library. Before starting to train the algorithm, an analysis of raw data is performed. Raw data is the data that comes to you directly from the task at hand. For example, this may be the personal data of the loan borrower. There will be strings with his name, work, maybe some numerical data. Another example of raw data is the search logs in the information retrieval system. Such logs are text strings that contain a unique identifier of the user, the time of his search, the request that interests him and the pages that he clicked on. All these examples are united by the fact that they do not explicitly present data in numerical form. This data must first be extracted from the raw data. And the first step in working with data is to extract it. The second stage is data transformation. It often turns out that even numerical data is not all suitable for machine learning algorithms. They may contain omissions and some kind of algorithm that we need can not work with omissions. We must somehow take this into account and transform the gaps in the data. In the case of the k-nearest neighbors algorithm, the model will be very sensitive to a large difference between the ranges of features’ values. If some attribute has values ​​from -1 to 1, and another attribute from -1 million to +1 million, then the algorithm simply will not take into account the first attribute due to the metric function used. We need some methods of processing features’ values ​​and in the pipeline of the intellectual data processing system there is a separate stage of analysis and processing of features - feature transformation. Only after this begins the stage of training the model, choosing the optimal parameters and comparing the model with other machine learning algorithms. When the model is selected, the final test of the model is performed before its release in production and, if the model shows good quality, then it can be used. Thus, it must be remembered that machine learning requires preliminary work with data and the choice of many models. Only after completing all the steps can we confidently use the machine learning algorithm in the production system. Now we can move on to the first steps indicated in this diagram.

Slide 22

And we will start by extracting the features’ from the raw data. I have already said that in many tasks we do not have the opportunity to directly access the data presented in the form of numbers. This can often be found in tasks such as text processing. In solving the problem of spam classification, we simply do not have any numerical data for training. We must invent them. Another example of a problem in which there is no suitable numerical data is the task of recommending the nearest restaurants. In this problem, there are both the coordinates of the person and the restaurants, as well as their descriptions. You cannot use the coordinates in raw form, you need to somehow convert them. Likewise, you cannot work with explicit textual descriptions of restaurants. Moreover, when we examine raw data to generate new features, we begin to better understand the problem, which means we increase the quality of the machine learning algorithm. We will briefly look at several types of raw data that you may encounter in a practical task.

Slide 23

The first type of raw data is text documents. In such data there are no numerical features at all. There are a large number of works devoted to the topic of extracting numerical features’ from textual data. We are not able to consider them all in the framework of this course, so we restrict ourselves to a brief description of what are the stages in word processing. At the first stage of working with texts, the text should be divided into sentences and words. This is called tokenization. After breaking the text into words, you may need to convert them to their normal form. This is true for languages ​​such as French, where the same word may have a different spelling depending on the context. In English, a word can change its shape depending on the time in the text. This step is called normalization. After him, all the words are reduced to a single form. Now you can work with text data. One way to convert text to a numerical format is to use the tf / idf algorithm. In this algorithm, each unique word is first assigned an index - this is a number in order. Then, for each word, its probability to be found in the document is calculated - this is the first part, called TF. Then, the value is considered that is the inverse of the logarithm of the fraction of documents that contains this word. This is the part called IDF. As a result, each word receives a numerical value that characterizes it for the document in question. All words are combined into one large vector consisting of tf / idf values ​​and this vector is used by the machine learning algorithm. If you want to learn more about the classical statistical methods of working with text, you can refer to the textbook of Jurafsky and Martin Speech & Language processing.

Slide 24

There are also methods that create new vector representations for text documents already converted to vectors. Now methods based on neural networks are very popular. The basis of these methods is the idea of ​​a “bag of words”, which represents a word as a set of counters of other words that surround it. The resulting counters are used by neural networks to convert them into small vectors. Such vector representations are then used by other machine learning algorithms to solve problems. If you are interested in getting to know these methods more closely, then the slide provides links to articles on some popular algorithms.

Slide 25

Another example of raw data is an image. On the one hand, an image is a matrix of numbers and they can be directly used as features in solving a machine learning problem. Indeed it is. But the machine learning algorithm will work much better if it is provided with additional information about the image. In particular, there are a large number of classical algorithms for extracting useful information from images. Such algorithms seek to clear the image of noise, highlight the boundaries of objects in the image, and identify specific image elements, such as angles or points. After that, these algorithms distinguish the objects themselves in the image and determine their features - where it is located and how in relation to other objects in the picture. After that, the features of the objects can be used to train the algorithm. If you want to know more about such algorithms, you can refer to the book by the link on the slide. However, in this case, neural networks are also popular in solving the problem of feature extraction.

Slide 26

The fact is that convolutional neural networks are arranged in such a way that they are able to automatically learn to perform most of the image analysis tasks that were solved by classical algorithms. The thing is that the convolutional neural network seeks to imitate the visual cortex of the human brain. This part of the brain is responsible for filtering noise and highlighting specific details in the image. The idea of ​​convolutional neural networks is to process only small sections of the image layer by layer, and then combine the information from them into something more. So roughly the human visual cortex also works. You can learn more about neural networks of this type from the link on the slide.

Slide 27

The next type of data worth mentioning is geodata, namely coordinates or address. Such data can be represented by a pair of numbers - latitude and longitude. On the other hand, this data can be represented by an address string. In the first case, it is necessary to keep in mind the accuracy of determining the coordinates of an object on the map using GPS. For example, with low accuracy in determining coordinates, a point on the map during movement can teleport from one place to another, which in real life is of course impossible. In the case of the address string, you need to keep in mind typos, as well as the fact that such an address must be geocoded. Geocoding is the process of determining coordinates at an address. Fortunately, there is an OpenStreetMap library that allows you to do this. But careful handling of geodata gives us a wealth of useful information. After processing such data, we get information about the nearest objects on the map, speed, and so on. All this information can greatly improve the quality of machine learning algorithms that solve problems that are tied to the physical location of the object on the map. For example, features extracted from coordinates can be useful in determining the cost of housing. Such features may be the proximity of shopping centers and kindergartens to the building, the presence of nearby park areas and so on.

Slide 28

Finally, another interesting source of data is date and time information. Such data are often found in the prediction of temporary delights. An example of such a task is predicting the load on cluster machines. When developing features based on timestamps, it should be borne in mind that data can be presented in a large number of formats. Only some of them are shown on the slide. And these formats need to be able to handle. Also note that comparing a date / time pair with another such pair is different from comparing only dates or times. Plus, do not forget that different countries have holidays and non-working days. For example, if you want to predict the sale of tickets to movie theaters, then it is worth considering that on weekends people often go to the movies. On holidays, people buy more products, and on weekdays people spend more time at their computer and don’t buy anything. All this information can provide a large amount of useful data for machine learning.

Slide 29

The next step after the numerical values ​​of the features are extracted from the data is the conversion of features. A feature transformation is an application to a feature of a function that replaces one numerical value with another with some additional properties while preserving the properties of the original feature. Why do we need to carry out such transformations? First, part of the features may have missing values ​​and it is often necessary to remove them from the data. Secondly, the features can differ greatly from each other in ranges of values, which can adversely affect the quality of work of a particular algorithm. So for example, the k-nearest neighbors algorithm suffers from this effect in the data. Finally, working with features and studying the distribution of data, we begin to better understand the properties of features and, as a result, the problem itself, and therefore increase the final quality of the solution.

Slide 30

The first transformation we look at is normalization. Normalization brings the range of values ​​of the attribute to a certain limited interval, well, or at least try to maximize narrow this interval. So, for example, z-scaling converts the trait into some semblance of a normal distribution with zero expectation and unit variance. Such a transformation will be very useful if there are outliers in the data because the main informative objects will be well grouped together, and the outliers will remain, but they will not get too much out of the data. Often the use of such a conversion is useful if you use the k-nearest neighbors algorithm. This transformation just solves the problem of a big difference in the ranges of feature’s values. Another conversion is MinMaxScaling. Such a transformation also changes the range of feature’s values ​​and leads it to a range from zero to one. Moreover, unlike the previous transformation, the distribution form does not change at all. Such a conversion is useful if you want to display the feature’s values ​​on the chart next to other features. In practice, z-scaling is still more often used, which leads the values ​​of the features to a distribution similar to a normal distribution. Often, algorithms work much better with attributes that are distributed in this way.

Slide 31

And it turns out that another useful transformation is the logarithm. The fact is that very often data can contain numbers that are very different from each other and at the same time are not outliers in the data. Then it is useful to apply logarithms, because it is a contraction transformation. It turns out that logarithm can also lead the data to a form similar to a normal distribution. On the slide you can see the Q-Q Plot graphs. Such graphs show how many numbers differ from the normal distribution. The farther the blue line is from the diagonal line in the chart, the smaller the data distribution is similar to normal. The slide shows the result of converting some attribute using the logarithm. As you can see, the values ​​that were strongly knocked out of the normal distribution are now slightly closer to the diagonal, which means that now the data distribution is closer to normal.

Slide 32

As you remember, there is an approach in machine learning on the generation of logical rules. So work, for example, decision trees. But what is a logical rule in a decision tree? This is a certain threshold function that associates 1 with certain ranges of values ​​of a feature. If the attribute does not fall into such a range, then the logical rule will return zero. In fact, we break the entire range of values ​​of the feature into ranges. Then, each value of the feature can be associated with the range number into which the feature falls. Thus, the operation of a logical rule is simulated, which denotes the interval in which the feature fell. In another way, such a transformation is called discretization. If for some reason we cannot use decision trees or we want to reduce excessive accuracy in the presentation of the feature, then using KBinsDiscretizer from the scikit learn library can help us with this.

Slide 33

But what if the feature is nominal or even a string? We still need to convert it to a numerical representation. If the number of variants of the attribute value is limited, then a vector consisting of N coordinates can be associated with the entire range of N possible values. With this transformation, the value of the attribute will be compared with its coordinate in the vector and the vector will receive the value 1 only in this coordinate. All other coordinates will keep the value zero. Thus we can use features that did not have before this numerical representation.

Slide 34

Often, exploration of the features leads to the idea that some features are useful to combine with each other to produce a new feature. The most important thing is to preserve the nonlinearity of such a transformation. So, for example, knowing the distance and the time during which the pedestrian traveled this distance, we can calculate its average speed. It is rare that a machine learning algorithm can automatically invent such a feature. But at the same time, if we solve the problem of predicting the time spent on the road, such a feature will be very important for us. After extracting the features from the raw data, this is the second place where a person can play a decisive role in increasing the quality of the solution. However, often features with non-linear transformations of the original features can simply be generated automatically by sorting out all sorts of conversion functions. For example, the PolynomialFeatures algorithm from the scikit learn library will simply perform a series of expansions to the degree of the initial features, which will, for example, turn a linear regression into polynomial.

Slide 35

Finally, as I said, we often need to process missing values ​​in the data. There can be many reasons for the omission - data noise, the high cost of performing measurements, human errors, and so on. Not all algorithms are capable of handling data gaps, unlike decision trees. How to replace the missing values ​​with some useful value? For this, scikit learn has an Imputer algorithm. The first method is the use of a simple replacement with the average value of this feature in the data or with the most frequent value. This replacement is very easy to calculate, but it completely ignores the fact that it averages objects of completely different classes. If there is a need to distinguish between different missing values ​​for different, then machine learning can be applied so that it predicts what value is possible instead of the missed one. For example, you can train a decision tree to fill in data gaps for all other feature’s values.

Slide 36

After the features have been extracted from the raw data and converted, you may need to select only part of the features for the task. What for? It would seem that the more features the better. But no. Firstly, not all features carry useful information and are involved in solving the problem. Such features can be dropped and the computational load on the computer reduced, which does not have to consider an extra feature. Secondly, uninformative features or features that look like noise in the data can spoil the training phase of the machine learning algorithm and reduce the quality of the final solution to the problem. Thirdly, when we automatically discard unnecessary features, it will be much easier for us to perform a manual analysis of the features to study the data. Finally, it is often useful to understand the task which features are more important and which are less important. For all this, there are methods of feature selection.

Slide 37

The most basic approach to selecting features is to use statistical criteria. So, for example, a feature with low variability, which is practically indistinguishable from a constant, can be thrown out of the sample because it does not contain useful information. Such filtering can be performed by the VarianceThreshold algorithm in the scikit learn library. Another approach is to calculate the numerical values ​​of the informativeness of the feature. A similar idea of ​​calculating information content works in algorithms based on logical rules. How does this approach work? We select a certain criterion of informativeness and sort by it all the features. After that, we select the first K most informative features. This is how the SelectKBest algorithm works.

Slide 38

Another interesting feature selection approach has appeared in linear machine learning algorithms. We considered the regularization of a linear model in which, as an additional restriction, the square of the length of the feature weight vector was added. It is interesting that if the square of the vector length is replaced by the sum of the absolute values ​​of the vector of weights, then such a regularization will automatically zero out the most uninformative features. In this way, we can use the linear machine learning model to determine the importance of individual attributes. In such a model, the resulting value of the feature’s weight will correspond to its importance. You can use decision trees. When using pruning in these algorithms, uninformative features will be filtered. Having trained such an algorithm, we can look at those features that participate in the final solution of the problem and determine which of them are the most informative.

Finally, we can just try to train the same model on different subsets of features and choose the most effective model. Thus, features that did not fall into this model can be considered uninformative.

Slide 39

Today we talked quite a bit about how machine learning algorithms work and discussed the technical issues of working with features and model training. Nevertheless, these are important topics since machine learning is always in the context of the existing practical task and we must understand how to proceed with solving the problem without having anything but raw data on hand. For this, we considered a machine learning pipeline, which consists of several stages, and these are: extracting features from raw data, transforming features, selecting features and models, training an algorithm, and finally, evaluating a machine learning algorithm. In the next lecture, we will touch upon another important concept in machine learning - the application of probability theory in machine learning.