Tic-Tac-Toe Endgame dataset

1 Name - Zolotareva Ekaterina

2 Neptun ID - DNBO45

3 Subject - Introduction to Data Science

4 Teacher name - Tomas Horvath

The task

The task was read, explore and perform Tic-Toc data from UCI repository, as well as to evaluate different algorithms which performs on the best way for Bianry Classification task.

Data understanding

The aim is collect initial data, get familiar with data and identify the quality of data.

This dataset encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row").

Interestingly, this raw database gives a stripped-down decision tree algorithm (e.g., ID3) fits. However, the rule-based CN2 algorithm, the simple IB1 instance-based learning algorithm, and the CITRE feature-constructing decision tree algorithm perform well on it.

- Data description:
  - Number of Instances: 958 (legal tic-tac-toe endgame boards)
  - Number of Attributes: 9, each corresponding to one tic-tac-toe square
  - Attribute Information: (x=player x has taken, o=player o has taken, b=blank)
  - top-left-square: x,o,b
  - top-middle-square: x,o,b
  - top-right-square: x,o,b
  - middle-left-square: x,o,b
  - middle-middle-square: x,o,b
  - middle-right-square: x,o,b
  - bottom-left-square: x,o,b
  - bottom-middle-square: x,o,b
  - bottom-right-square: x,o,b
  - Class: positive,negative
  - Missing Attribute Values: None
  - Class Distribution: About 65.3

In the UCI documentation mentioned few experiment results which were observed used 70 percent for training, 30 percent of the instances for testing, evaluated over 10 trials.

Our data:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	x	x	x	x	o	o	x	o	o	positive
1	x	x	x	x	o	o	o	x	o	positive
2	x	x	x	x	o	o	o	o	x	positive
3	x	x	x	x	o	o	o	b	b	positive
4	x	x	x	x	o	o	b	o	b	positive

When we get a description of the data we see that there are no NaN values in the dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 958 entries, 0 to 957
Data columns (total 10 columns):
V1     958 non-null int64
V2     958 non-null int64
V3     958 non-null int64
V4     958 non-null int64
V5     958 non-null int64
V6     958 non-null int64
V7     958 non-null int64
V8     958 non-null int64
V9     958 non-null int64
V10    958 non-null int64
dtypes: int64(10)
memory usage: 74.9 KB
```

Data preparation

The aim is to construct the final dataset from raw data which will be input for the modeling tool.

We change object data type to numerical data type:

- x - 0
- o - 1
- b - 2
- positive - 1
- IB3-CI: 99.1 %
- negative - 0

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	1	1	0	1	1	1
1	0	0	0	0	1	1	1	0	1	1
2	0	0	0	0	1	1	1	1	0	1
3	0	0	0	0	1	1	1	2	2	1
4	0	0	0	0	1	1	2	1	2	1

And also splitted our data on X (design matrix) and y (target).

Modeling

Defining the problem

Among all Machine Learning problems we can highlight the most basic:

- classification of an instance to one of the categories based on its features;
- regression - prediction of a numerical target feature based on other features of an instance;
- clustering - identifying partitions of instances based on the features of these instances so that the members within the groups are more similar to each other than those in the other groups;
- anomaly detection - search for instances that are "greatly dissimilar" to the rest of the sample or to some group of instances;
- and so many more.

First of all we need to define which task we have and only after think about approaches and another problems. We will remind the task is: to read, to explore and to perform Tic-Toc data from UCI repository, as well as to evaluate different algorithms which performs on the best way for Bianry Classification task. Consequently, we need to predict if the specified combination of walks will allow us to win or reversely, not. This kind of problem is usual task of binary classification.

We defined the task and it is Binary Classification problem. Precisely, because our predicted target class ($\hat{y}$) will consist of two values: 1 - positive and 0 - negative.

Binary or binomial classification is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to) on the basis of a classification rule.

Selection of algorithms

Due to the fact that we have the usual problem of Binary Classification, I decided to try the standard algorithms of machine learning.
In my solution I tried the following algorithms:

- Support Vector Classification
- Logistic Regression
- KNeighbors Classification
- Random Forest Classification
- Gradient Boosting Classification

"Defining" the evaluating metrics

Speaking about the estimation metrics of the algorithm in the problem of the Binary classification, we can single out the main metrics.

Before moving to the metrics themselves, it is necessary to introduce an important concept to describe these metrics in terms of classification errors - the confusion matrix. Suppose that we have two classes and an algorithm that predicts that each object belongs to one of the classes, then the classification error matrix will look like this:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Classification errors are of two types: False Negative (FN) and False Positive (FP).

- Accuracy - intuitive, obvious and almost unused metric is accuracy - the proportion of correct answers of the algorithm. This metric is useless in problems with unequal classes.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Precision and Recall - To assess the quality of the algorithm's work on each of the classes separately, we introduce precision (accuracy) and recall (complete) metrics.

$$Precision = \frac{TP}{TP+FP}.$$

$$Reacall = \frac{TP}{TP+FN}.$$

  Precision and recall do not depend, as opposed to accuracy, on the class ratios and are therefore applicable in conditions of unbalanced samples.

  There are several different ways to combine precision and recall into an aggregated quality criterion.

Machine Learning approaches

We will use GridSearchCV from sklearn for searching best model with best set of parameters.

The following parameters will be passed:

- estimator = model
- param grid = grid for current model
- n jobs = -1
- cv = KFold(nsplits=10, randomstate = 1)
- refit = accuracy
- scoring = accuracy, recall, precision
- error score = 0

Support Vector Classification

The following parameters are submitted to the model input:

- C - Regularization parameter : $[1, 10]$
- $kernel$ - Specifies the kernel type to be used in the algorithm : $linear, rbf$

We have next output:

- best accuracy: 0.954070981210856
- best precision: 0.9734112327014206
- best recall: 1.0

Logistic Regression

The following parameters are submitted to the model input:

- solvers = [newton-cg, lbfgs, liblinear]
- penalty = l2
- cvalues = [100, 10, 1.0, 0.1, 0.01]

After define models and parameters and grid search we summarized results and have:

- best accuracy: 0.5146137787056367
- best precision: 0.652423044463199
- best recall: 0.5607098121085594

KNeighbors Classification

The following parameters are submitted to the model input:

- nneighbors = range(1, 21, 2)
- weights = [uniform, distance]
- metric = [euclidean, manhattan, minkowski]

After define models and parameters and grid search we summarized results and have:

- best accuracy: 0.7108559498956158
- best precision: 0.6638830897703549
- best recall: 0.6284759916492693

Random Forest Classifier

The following parameters are submitted to the model input:

- nestimators = [10, 100, 1000]
- maxfeatures = [sqrt, log2]

After define models and parameters and grid search we summarized results and have:

- best accuracy: 0.6085594989561587
- best precision: 0.671988210733145
- best recall: 0.558455114822547

Gradient Boosting Classifier

The following parameters are submitted to the model input:

- nestimators = [10, 100, 1000]
- learningrate = [0.001, 0.01, 0.1]
- subsample = [0.5, 0.7, 1.0]
- maxdepth = [3, 7, 9]

After define models and parameters and grid search we summarized results and have:

- best accuracy: 0.8705636743215032
- best precision: 0.6718221647211032
- best recall: 0.7014613778705637

Comparison of evaluation

	model	accuracy	precision	recall
0	SVC	0.954070981210856	0.9734112327014206	1.0
1	Logistic Regression	0.5146137787056367	0.652423044463199	0.5607098121085594
2	KNeighborsClassifier	0.7108559498956158	0.6638830897703549	0.6284759916492693
3	RandomForestClassifier	0.6085594989561587	0.671988210733145	0.558455114822547
4	GradientBoostingClassifier	0.8705636743215032	0.6718221647211032	0.7014613778705637