Mushroom Data Set
Introduction to Machine Learning, Project 1

Julius Elfving, Piotr Rokicki, Karin Frlic

October 10, 2017

1 Introduction

This is a report about the supervised learning project on course Introduction to machine learning. We chose to use a dataset about mushrooms and the purpose was to classify them into poisonous and edible mushrooms. The dataset is described in this introduction part.

Link to the Mushroom data set can be found in part 6: Resources (Resource 1).

Mushroom data set has 8124 records. The records are divided into two classes which are poisonous (48.2% of records) and edible (51.8% of records). There are 22 attributes which are all nominal. Each record represents one mushroom and attributes describe what kind of mushroom it is.

Attribute description:

All the attributes and their possible values are listed below. There are 2480 (30.5% of records) missing values in attribute 11, stalk-root. For other attributes there are no missing values. The distribution of data can be seen in a figure in part 2.1.

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

Record example:

An example of what the data looks like. This is the first record of the data set.

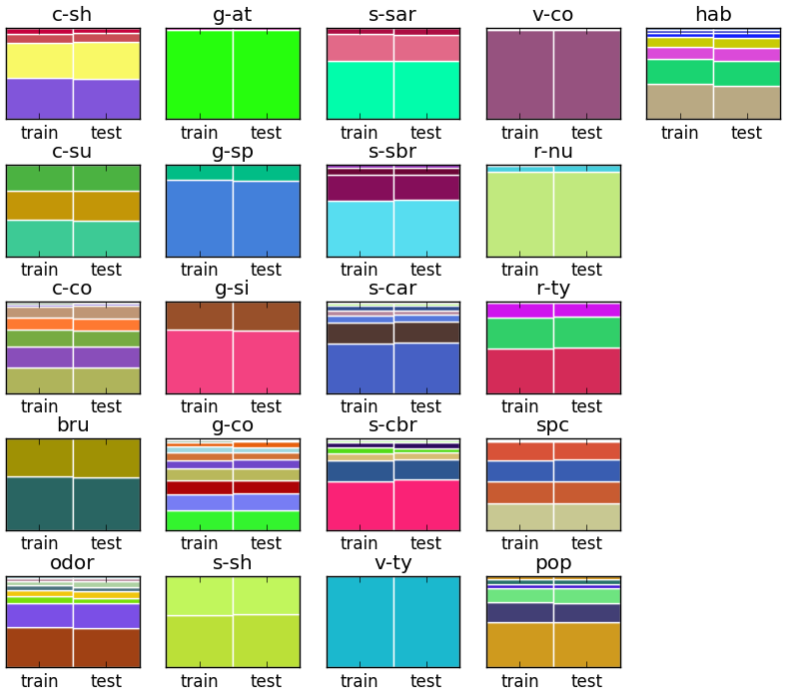
```
cl       p
c-sh     x
c-su     s
c-co     n
bru      t
odor     p
g-at     f
g-sp     c
g-si     n
g-co     k
s-sh     e
s-ro     e
s-sar    s
s-sbr    s
s-car    w
s-cbr    w
v-ty     p
v-co     w
r-nu     o
r-ty     p
spc      k
pop      s
hab      u
```

2 Process

2.1 Splitting the data to train and test set

We decided to use 80% of the data to train the classifier and 20% to test. We checked the distributions of attributes and classes in both training set and test set and found out that attribute values and classes are distributed well.

Attribute distribution:



Class distribution:

2.2 Selecting models

We decided to use five classifiers:

1. **k-nearest neighbours**
   We thought that mushrooms that are poisonous have many common characteristics so they would be close together, and that the same goes for eadible ones. On the other hand, poisonous and edible mushrooms should be less similar to each other so the distance between them should be larger.

2. **Bayes**
   Poisonous mushrooms should have a set of attributes' values that can tell you that they are not edible while edible mushrooms should have different values for those attributes, so we thought that this classifier could work well.

3. **Support vector machines**
   In the same fashion, we predicted that the differences are obvious so there exists a hyperplane that can separate two classes fairly well.

4. **Decision tree**
   We expected that we don't need all attributes to classify the records so we decided to use the decision tree classifier to find out which attributes are the most important for classification.

5. **Neural networks**
   In the end, we decided to try using a neural networks to see how well it works.

2.3 Preprocessing

Some records don't have the values for attribute *stalk-root*. We decided to ignore it since there are enough of other attributes.

All the attributes are nominal (characters), so we first had to transform them to numbers - we used *LabelEncoder*. Because that put some order in our data set, we used *OneHotEncoder* to transform the data in a way that now all attributes are boolean (have either value 0 or 1) and each one represents a combination *(attribute, value)* - the value of that new attribute is 1 if the original *attribute* has value *value* and 0 otherwise.

2.4 Hyperparameters

1. **k-nearest neighbours**
   We wanted to see if it is enough to look only at one nearest neighbour or if we need more of them to classify correctly. On the other hand, we thought that 100 neighbours should be enough since we predicted the differences between classes are big.
2. **Bayes**
   We decided to test how well does the classifier perform using different values for *alpha* (Laplace/Lidstone smoothing parameter) from the range [0, 1] because we have read (Resource 2) that the values chosen for that parameter are normally in this range.
3. **Support vector machines**
   Only a linear kernel function was used because there were convergence problems with other kernels. However, linear kernel function was working fine, so there was no need for other kernels. For other parameters we decided to test the default values and add some other numbers that seem reasonable.

4. **Decision tree**
   For parameter *max_features* we predicted that good results can be achieved without considering all features each time we split the node, so we planed to focus on smaller percentages but also add some slightly bigger.
5. **Neural networks**
   Reasons for choosing particual values for each of the parameters of Neural Network vary: maximal number of iterations, type of learning rate and its initial value, the momentum, all these affect the speed of the learning process (and how long do we allow it to learn) as well as the network vulnerability (or not) to being stuck in local minima (e.g. too low momentum) or not being able to properly converge (e.g. too big constant learning rate). On the other hand, although the size of the hidden layers, the type of activation funtion and the solver affect these things as well, they also affect the very ability of given network to solve given problem. Incorrect activation function or too small hidden layer may render the network completelly useless. Sigmoid function was chosen due to it being the most popular, widely used and able to tackle many problems, "adam" and "sgd" solvers were chosen due to dataset being quite big ("adam" is best for that (Resource 3), "sgd" is just a standard) and range of sizes of hidden layers were chosen more freely as with problem of this complexity there is no need for humongous hidden layer but also there is not that much data to actively try to reduce the size of hidden layer due to time consuming learning process.

2.5 Evaluation

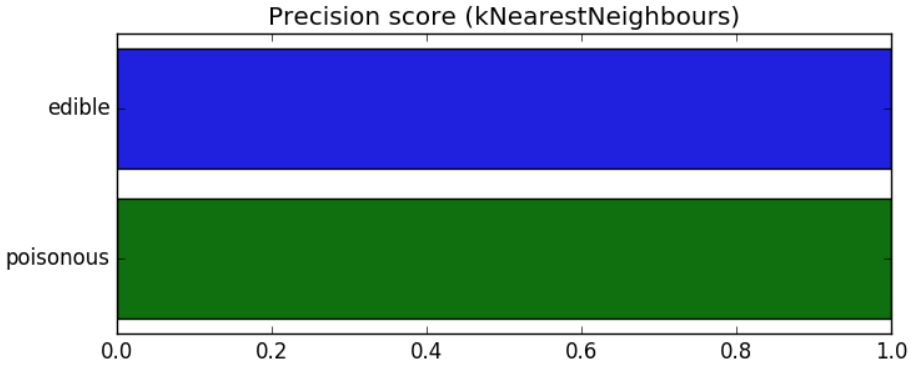
To choose the best model, we planed to compare reports produced by *GridSearchCV*, which performs k-fold (3-fold) validation and then select the one with the highest precision and recall score.

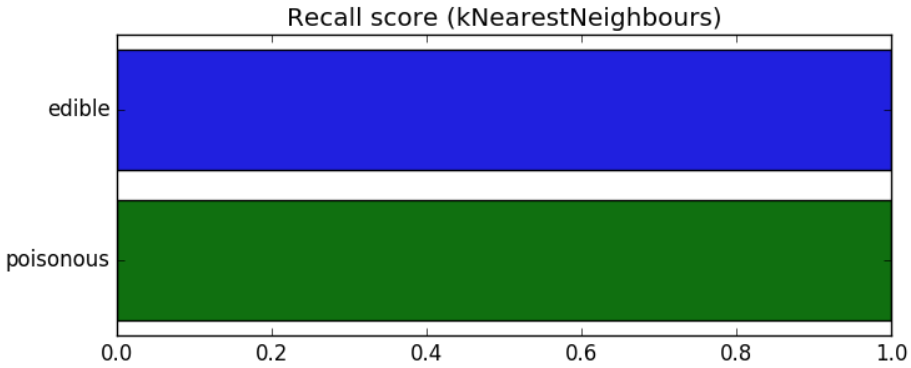
3 Results

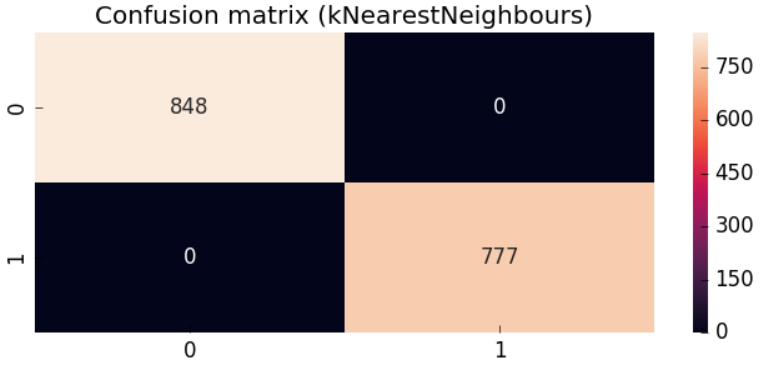
Results in form of plots can be found below. For each classifier there are precision score, recall score and confussion matrix plots.

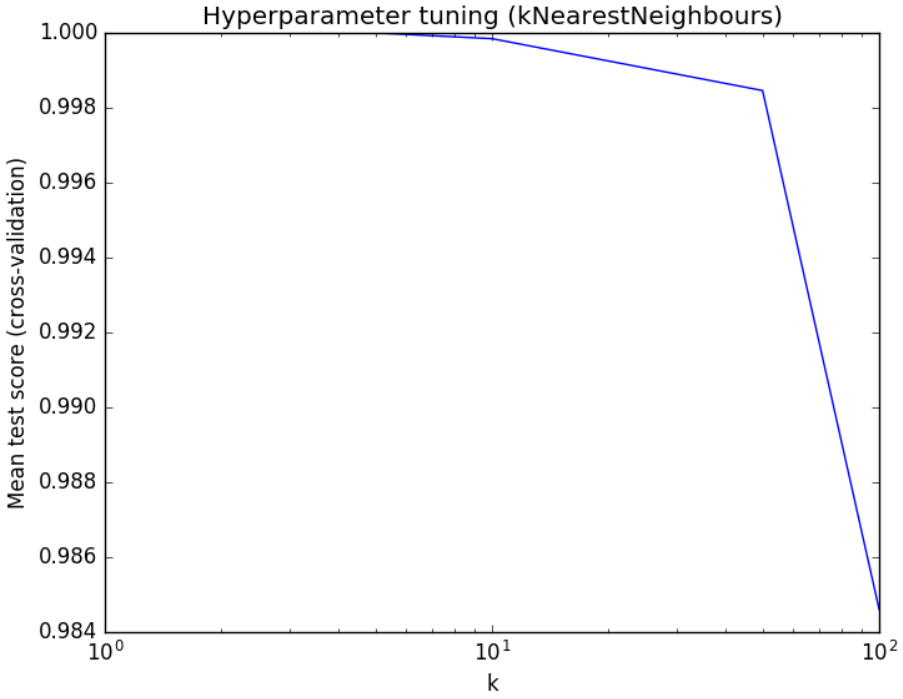
3.1 KNN

Hyperparameters chosen:
n_neighbors = **1**





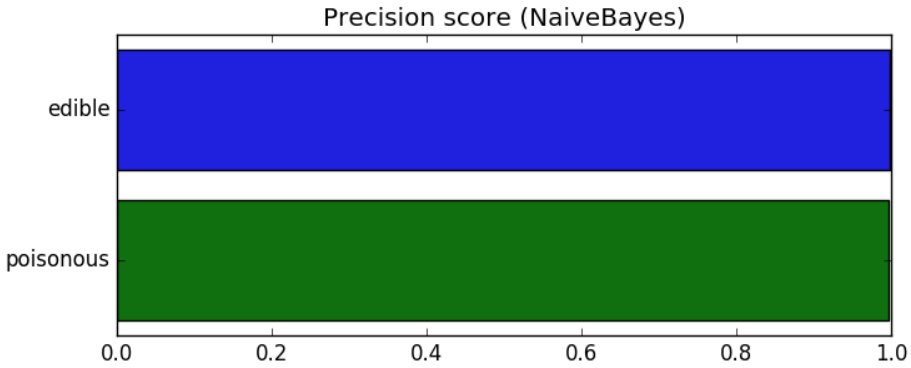


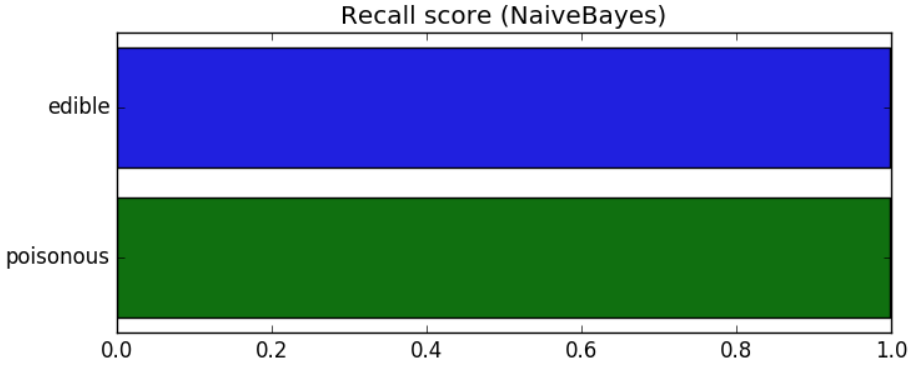


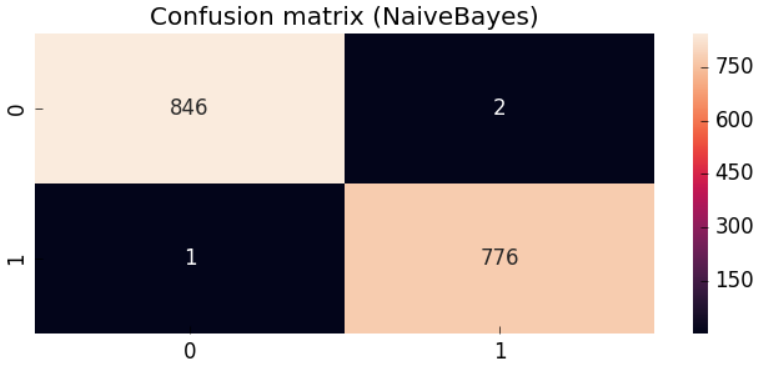
The last plot shows how the value of hyperparameter k affects the overall score.

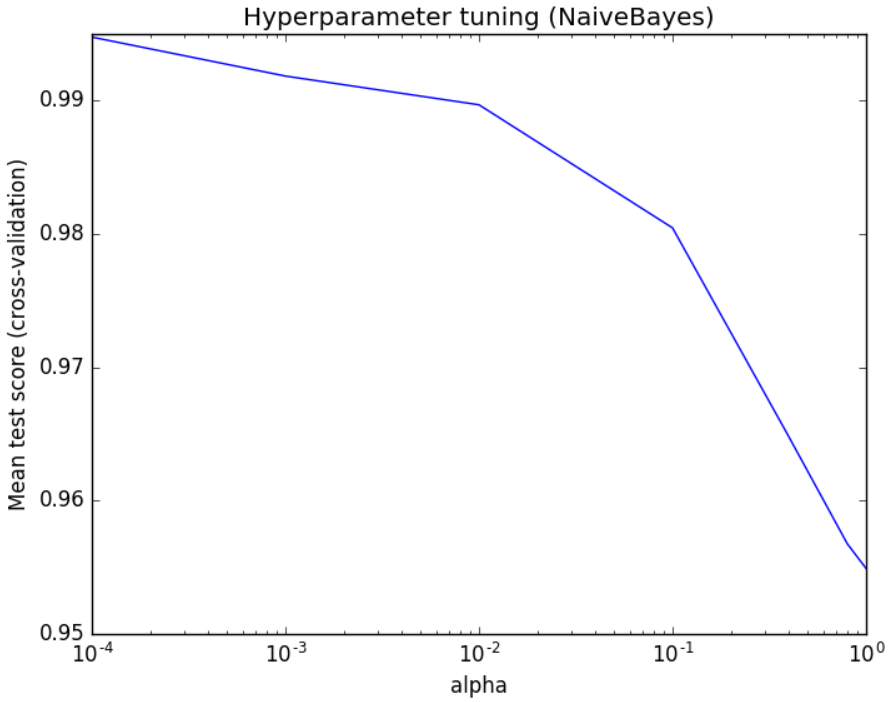
3.2 Bayes

Hyperparameters chosen:
alpha = **0.0001**


Precision score (NaiveBayes)
edible
poisonous
0.0
0.2
0.4
0.6
0.8
1.0


Recall score (NaiveBayes)
edible
poisonous
0.0
0.2
0.4
0.6
0.8
1.0

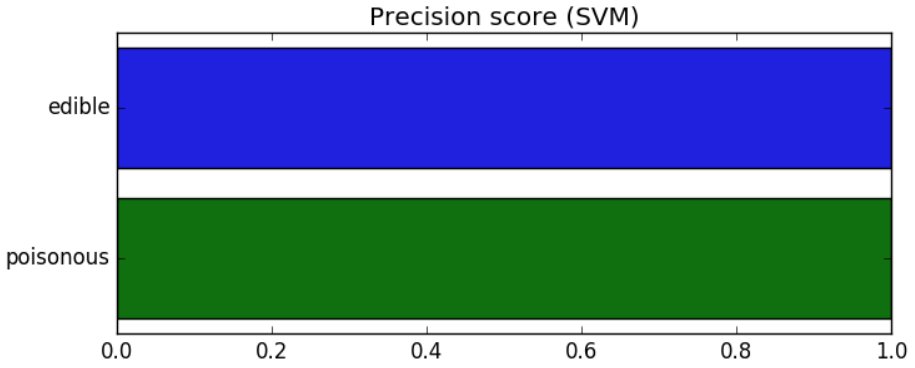

Confusion matrix (NaiveBayes)
846
2
1
776
0
1
0
1
750
600
450
300
150

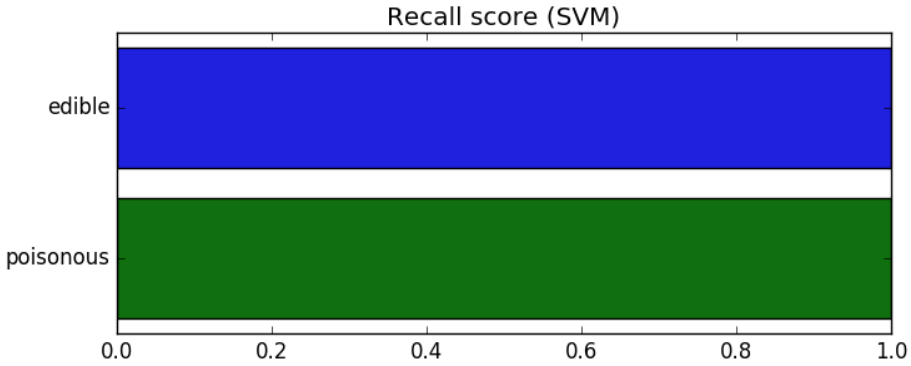


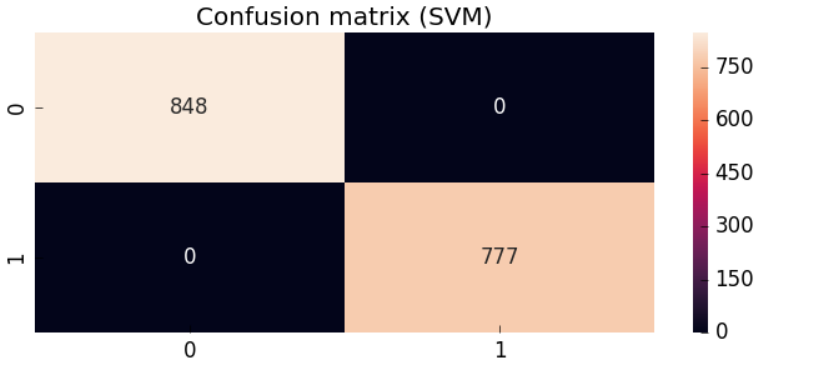
The last plot shows how the value of hyperparameter alpha affects the overall score.

3.3 SVM

Hyperparameters chosen:
kernel = **linear**
decision_function_shape = **ovo**
gamma = **0.001**
coef0 = **0.001**

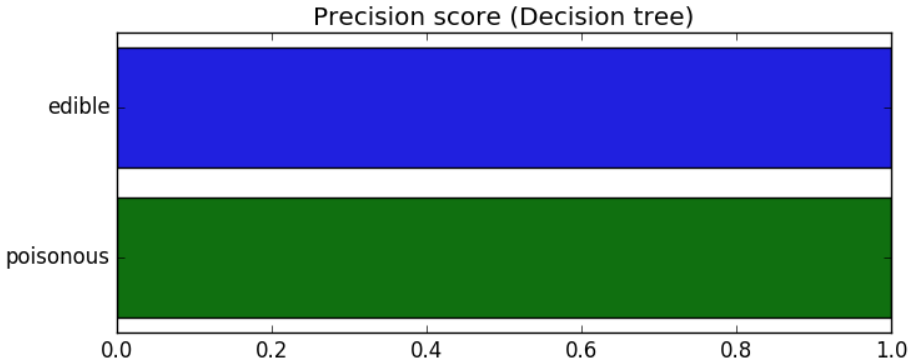


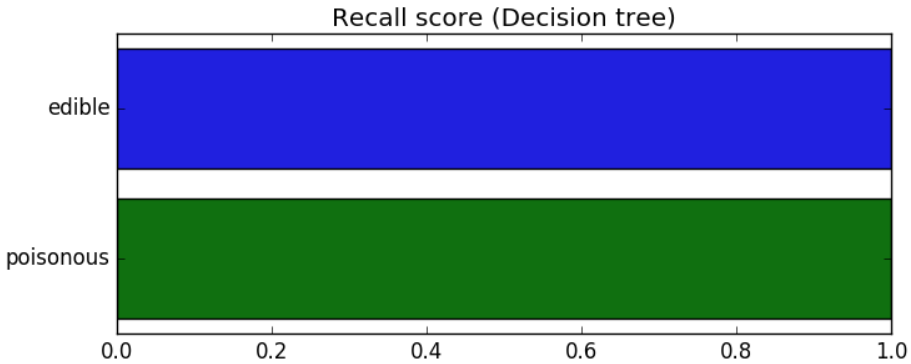


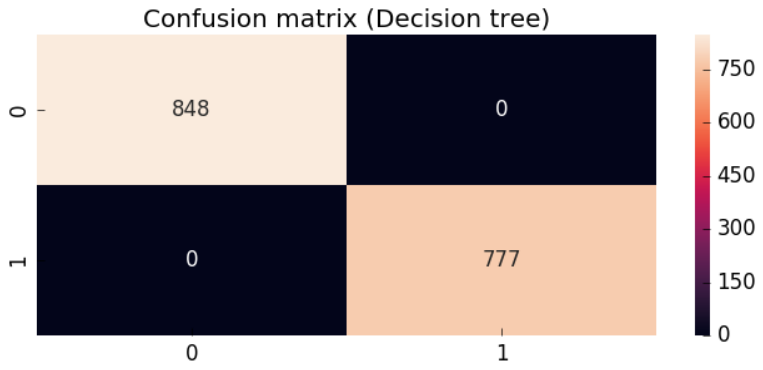


3.4 Decision Tree

Hyperparameters chosen:
min_samples_leaf = **3**
max_features = **0.35**

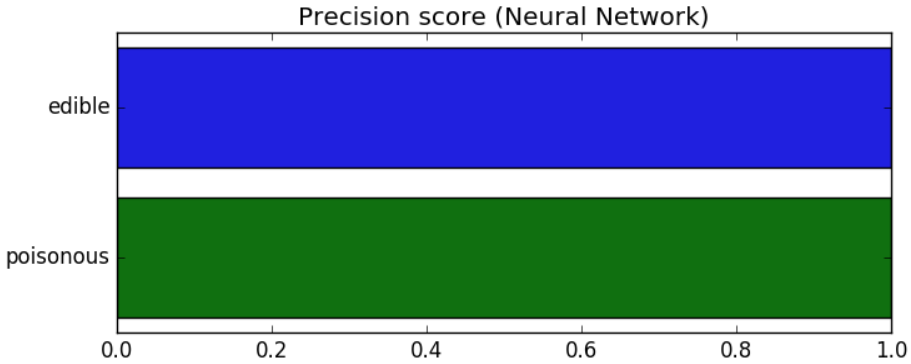


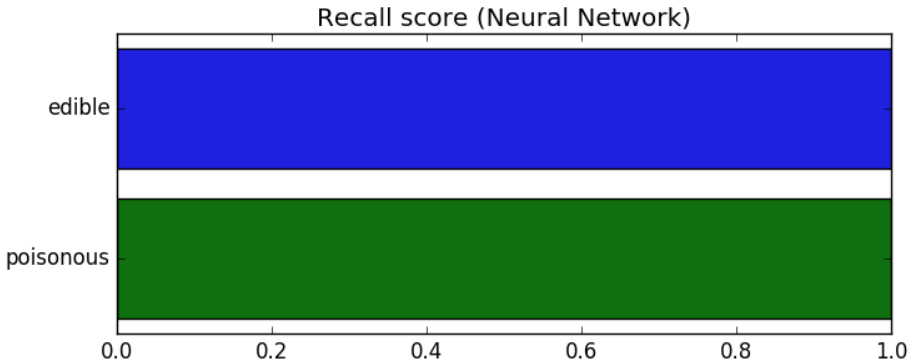


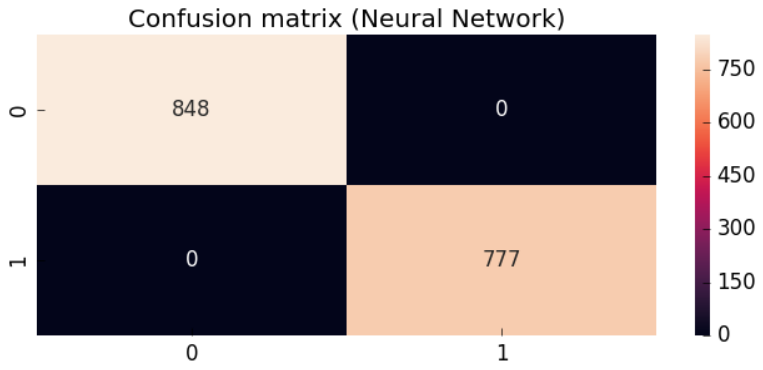


3.5 Neural Networks

Hyperparameters chosen:
activation = **logistic**
hidden_layer_sizes = **(50, 50)**
solver = **adam**
learning_rate = **constant**
momentum = **0.5**
max_iter = **2000**
learning_rate_init = **0.1**

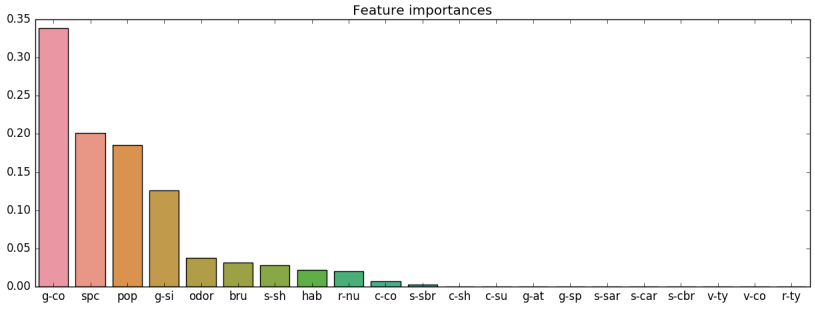






3.6 Feature importance

There is a plot of *Feature importance* below, which means, how much each of the attributes affect the final decision (edible - poisonous). Achieved with Decision tree. These results were not used in this project but are discussed in part 5 Future work.



3.7 Comparing different models

To summarize the results shown above, there is a table of the results got using different classifiers. There are two scores for each classifier, minimun score and maximum score which are both mean test scores when using cross-validation. Minimun score is the mean test score when using the worst hyperparameter values. Maximum score is the mean test score when using the best hyperparameter values that were found. Combinations is the number of different hyperparameter combinations that were tested.

Classifier	Min score	Max score	Combinations
Decision tree	0.998	1.000	12
NaiveBayes	0.955	0.995	7
Neural Network	0.517	1.000	72
SVM	1.000	1.000	60
kNearestNeighbours	0.985	1.000	6

4 Conclusions

All the models that were tested resulted a test score of over 99% which makes it difficult to compare different classifiers. The fact that all models gave such good results shows that the mushroom data set is fairly easy to classify correctly.

4.1 Reference reports

There are a few informal reports of classifying this data set. Whitney had used an artificial neural network to classify mushrooms and got 99.1% of the records classified correctly (Resource 4). Ranthony had used Naive Bayes and got mean accuracy of 94.2% (Resource 5). Haunschmid had also used Naive Bayes and got 94.0% of the records classified correctly (Resource 6). The referred reports aren't very formal but they give some idea of what could the accuracy be.

The difference in results can be explained with different preprocessing or with different hyperparameter tuning. In (Resource 4) which used neural network, one hot encoding is not used, meaning that the attributes are not considered nominal but they have an order. That's one mistake in the preprocessing that could lead to a different result. (Resource 5) and (Resource 6) both used Naive Bayes and achieved an accuracy of approximately 94%. Naive Bayes seemed to give results of 95% when using bad hyperparameter values so the difference in results could be explained with that. Also removing the attribute that had missing values could have improved the accuracy.

4.2 Hyperparameters

Each of the classifiers that were used had some hyperparameters that could be varied. From the results it can be seen that Support vector machine was the only classifier that got 100% accuracy independent of the hyperparameters. Decision tree and k nearest neighbors did a bit worse with bad hyperparameters values resulting 99.8% and 98.5%, respectively. For Naive Bayes there was clear difference between the best and the worst hyperparameter values. Mean test scores for cross-validation were 99.5% for the best hyperparameters and 95.5% for the worst hyperparameters. Naive Bayes was also only classifier to not achieve accuracy of 100% using the hyperparameter values that were tested.

Biggest difference in scores between hyperparameters could be seen when using Neural network classifier. There with an unsuccessful hyperparameter selection the score could drop to only 51.7% which is a terrible accuracy. The accuracy of 51.7% could mean that the model classifies every record as edible because 51.8% of the records belong to class edible. 72 different hyperparameter combinations were tried for neural network and only five of them gave an accuracy worse than 98% so most of the hyperparameter selections worked well.

For this data set, it wasn't a hard job to find good values for hyperparameters. However, from the example of neural networks it can be seen, that choosing wrong hyperparameters can lead to terrible results. Maybe it could be possible to select very bad hyperparameter values for also the other classifiers but in this case it was easy to find good values.

5 Future work

The selected data set turned out to be fairly easy to classify. Because the accuracy of 100% was obtained there is no need to try to find a better model. However, some other analyses could be done on this dataset, for example using only a few most important attributes (see *Feature importance* plot) for classification and seeing how well it works.

We have learnt that it is hard to choose the best model for selected data set because the majority of classifiers that we have tried always predicted the right output. One way to select which one performs best would be to measure the time needed for classification and select the fastest one. On the other hand, we could compare them according to space the model needs and select the one that is the most space-eficient.

6 Resources

1. "Mushroom Data Set". UCI Machine Learning Repository. Retrieved September 30, 2017 from https://archive.ics.uci.edu/ml/datasets/Mushroom
2. "Additive smoothing". Wikipedia. Retrieved October 1, 2017, from https://en.wikipedia.org/wiki/Additive_smoothing
3. "MultinomialNB". Scikit-learn: Machine Learning in Python. Retrieved October 5, 2017 from http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
4. "Application: The Classic Mushroom Dataset". Retrieved October 9, 2017, from http://trevorwhitney.com/data_mining/mushrooms
5. "Mushrooms Data Set". Retrieved October 9, 2017, from https://rstudio-pubs-static.s3.amazonaws.com/169791_2924e5aa237549ae8a9af2ddabfebaa5.html
6. "A05_naivebayes". Retrieved October 9, 2017, from https://github.com/expectopatronum/ds-learning-club/blob/master/05-naivebayes/A05_naivebayes.pdf