katta. Nutqni tanib olish bugungi kunda eng muhim dasturlardan biridir. Taniqli bosma yoki qo‘lda yozilgan OCR yozib olinishi va nutq chiqishi mumkin edi. Bu ko‘zi ojizlarga ma’lumot yuborish va qabul qilishga yordam beradi.

Foydalanilgan adabiyotlar ro‘yxati:

1. Choryorqulov G‘.H., & Qosimov N.S. (2023). ELEKTRON JADVAL MODELINING TAVSIFLANISHI. PEDAGOGS Jurnali, 30(3), 67–73.
2. TA’LIMDA DASTURLASH JARAYONINI BAHOLASHGA ASOSLANGAN AVTOMATLASHTIRILGAN TIZIMNI TADBIQ ETISH Normatov N.K., Choryorqulov G‘.H., Zamonaviy innovatsion tadqiqotlarning dolzarb muammolari va rivojlanish tendensiyalari: yechimlar va istiqbollar mavzusidagi Respublika ilmiy-texnik anjumani-2023, 20-24-betlar.
3. Javlon X. et al. Классификатор движения рук с использованием биомиметического распознавания образов с помощью сверточных нейронных сетей с методом динамического порога для извлечения движения с использованием датчиков EF //Journal of new century innovations. – 2022. – Т. 19. – №. 6. – С. 352-357.
4. Юсупович Ҳ. Ж., Эргашев С. Б. Ў. МАКТАБ ЎҚУВЧИЛАРИДА АХБОРОТ БИЛАН ИШЛАШ КОМПЕТЕНЦИЯСИНИ РИВОЖЛАНТИРИШИ МОДЕЛИ //JOURNAL OF INNOVATIONS IN SCIENTIFIC AND EDUCATIONAL RESEARCH. – 2022. – Т. 2. – №. 13. – С. 189-194.
5. Obid o‘g A. S. J. et al. Numpy Library Capabilities. Vectorized Calculation In Numpy Va Type Of Information //Eurasian Research Bulletin. – 2022. – Т. 15. – С. 132-137.
6. Tavboyev Sirojiddin Akhbutayevich, Mamaraimov Abror Kamoliddin ugli, and Karshibaev Nizomiddin Abdumalikovich, “Algorithms for Selecting the Contour Lines of Images Based on the Theory of Fuzzy Sets”, TJET, vol. 15, pp. 31–40, Dec. 2022.

DECISION TREE CLASSIFICATION IN MACHINE LEARNING AND HYPERPARAMETERS

Salimov Jamshid Obid o‘g‘li
Jizzakh branch of National University of Uzbekistan
Abylayeva Akbota Muhamediyarovna
Eurasian National University named after L.N. Gumilyov
jamshidsalimov8@gmail.com

Annotation: Machine learning algorithms play a crucial role in extracting valuable insights from data, enabling businesses and researchers to make informed decisions. One such algorithm is the decision tree, which is widely used for classification tasks. Decision tree classification utilizes a tree-like model of decisions and their potential consequences, making it an intuitive and powerful tool for solving complex problems. In this article, a model that determines which drug is suitable for a patient with a certain disease is created using the Decision tree algorithm. This problem is multi-class classification (multiclass classification) help score consolidation. Alternatively, how function, domain, and hyperparameters simplify decision tree models are explored.

Key words: model, dataset, test set, training set, hyperparameters, classification, prediction, decision tree, multiclass classification.

The purpose of decision tree classification is to divide a dataset into homogeneous groups based on input features, leading to the assignment of categorical labels to new, unseen instances. This algorithm mimics human decision-making processes by forming a tree structure where each internal node represents a decision based on a feature, and each leaf node represents a class label.

Decision trees are versatile and find application in various domains. They are widely used in finance for credit scoring, fraud detection, and risk assessment. In healthcare, decision trees aid in disease diagnosis and treatment prediction. Additionally, decision trees are valuable in customer segmentation, sentiment analysis, and recommendation systems.

The decision tree algorithm involves a series of steps to construct an optimal tree structure. It begins with selecting the best feature from the dataset that effectively divides the instances into distinct classes. This process is repeated recursively for each subset of instances until the tree is fully grown. The algorithm uses measures such as information gain, gain ratio, or Gini index to evaluate the feature's effectiveness at splitting the data.

Hyperparameters are settings that control the behavior of the machine learning algorithm. In the context of decision trees, hyperparameters influence the tree's structure and complexity. Tuning these hyperparameters can simplify the decision tree model and improve its performance.

One common hyperparameter is the maximum depth of the tree, which limits the number of decision nodes and reduces complexity. Lowering the maximum depth helps avoid overfitting. Another crucial hyperparameter is the minimum number of samples required to split an internal node. Increasing this value prevents the creation of small branches that may lead to overfitting.

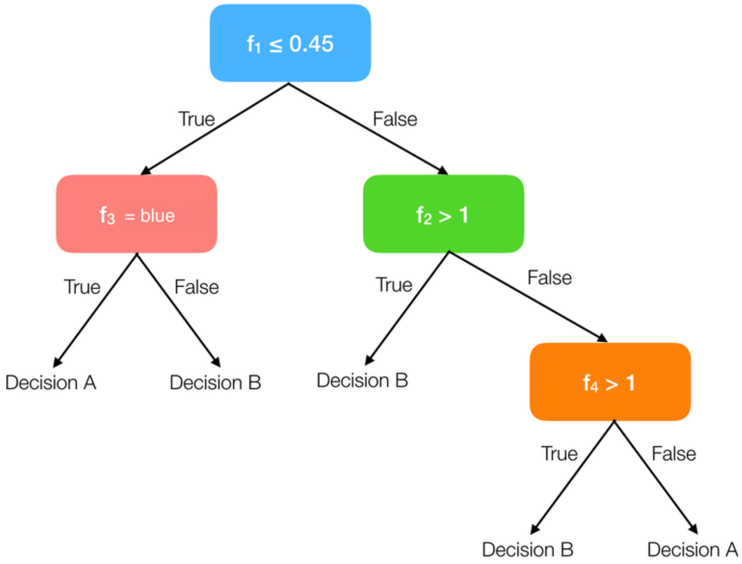
Additionally, decision tree models can be regularized using hyperparameters like minimum impurity decrease or maximum number of leaf nodes. These parameters control the growth of the tree by setting thresholds for stopping the splitting process.

Decision Tree Algoritmi

Medical data is collected for research. This data (DataSet) is about patients suffering from the same disease. During the course of treatment, one of the 5 different drugs was given to the patient.

The goal is to create a model using the Decision tree algorithm that determines which drug may be suitable for a future patient with the same disease. This problem is solved using multiclass classification.

The graphic below is an example of how a Decision Tree works.



How to build a Decision Tree model

- Select a column of the dataset
- We consider column importance in data partitioning
- We split the data by the best column
- We repeat the above steps.

Entropy determines which division leads to a better result.

$$E = -\sum_{i=1}^{n} p_i \, log_2(p_i)$$

https://journal.jbnuu.uz/index.php/ijcstr/article/view/579

The necessary libraries and modules for the Decision Tree algorithm are called.

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn import tree
from matplotlib import pyplot as plt
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

The required dataset is called to build the model.
https://raw.githubusercontent.com/JamshidSalimov/Ai-Fayls/master/drug200.csv

```
df=pd.read_csv("https://raw.githubusercontent.com/JamshidSalimov/Ai-Fayls/master/drug200.csv")
df
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

Textual data is converted to digital form.

```
encoder = LabelEncoder()
df['Sex'] = encoder.fit_transform(df['Sex'].values)
df['BP'] = encoder.fit_transform(df['BP'].values)
df['Cholesterol'] = encoder.fit_transform(df['Cholesterol'].values)
df.sample(5)
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
110	50	1	0	0	7.490	drugA
104	22	1	0	1	28.294	drugY
140	49	1	0	1	6.269	drugA
142	60	1	0	1	8.621	drugB
87	69	1	1	0	15.478	drugY

The predictor and original values are extracted from the dataset and equated to the X and y variables

```
X = df[['Age','Sex','BP','Cholesterol','Na_to_K']].values
y = df['Drug'].values
print(X[0:5])
print(y[0:5])
```

```
[[23.     0.     0.     0.     25.355]
 [47.     1.     1.     0.     13.093]
 [47.     1.     1.     0.     10.114]
 [28.     0.     2.     0.      7.798]
 [61.     0.     1.     0.     18.043]]
['drugY' 'drugC' 'drugC' 'drugX' 'drugY']
```

Dataset is split into train_set and test_set using train_test_split() module (60% train_set, 40% test_set)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=20)
```

The Decision Tree module is called and trained based on the Dataset.

```
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

Predictive values are found

```
# predicted values
y_predict = tree_model.predict(X_test)
```

Based on the predicted values, the model is evaluated

Model performance is evaluated using the classification_report() module based on actual values and predicted values

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

       drugA       1.00      1.00      1.00         7
       drugB       1.00      1.00      1.00         4
       drugC       0.75      1.00      0.86         9
       drugX       1.00      0.81      0.89        21
       drugY       0.97      1.00      0.99        39

    accuracy                           0.95        80
   macro avg       0.94      0.96      0.95        80
weighted avg       0.96      0.95      0.95        80
```

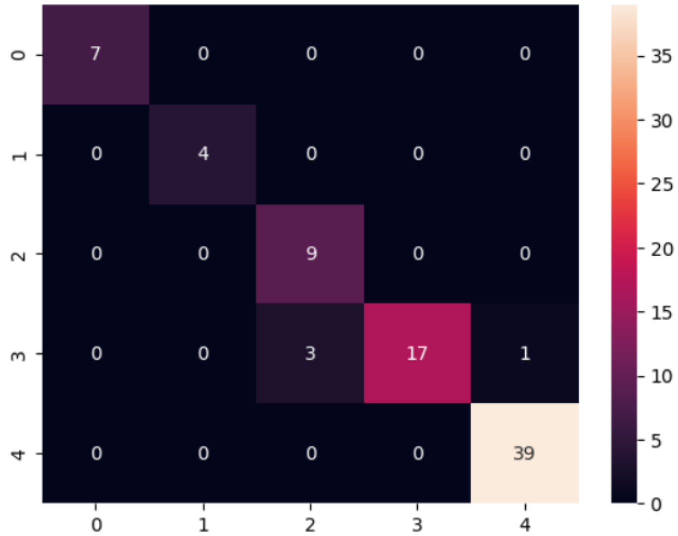
It is also possible to calculate these values separately

```
print("DecisionTrees's Accuracy: ", accuracy_score(y_test, y_predict))
```

```
DecisionTrees's Accuracy:  0.95
```

It is also possible to evaluate in matrix form using the confusion_matrix() module.

```
sns.heatmap(confusion_matrix(y_test, y_predict), annot=True)
plt.show()
```

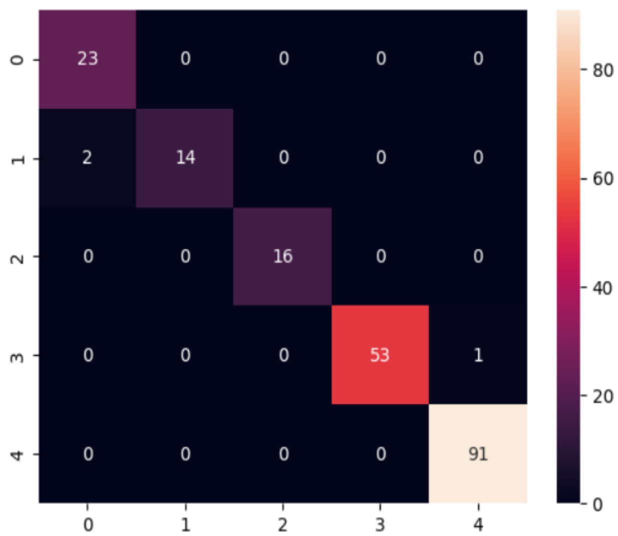


Cross-validation of the model may give better results

```
predict = cross_val_predict(estimator = tree_model, X = X, y = y, cv = 5)
print("Classification Report: \n",classification_report(y, predict))
```

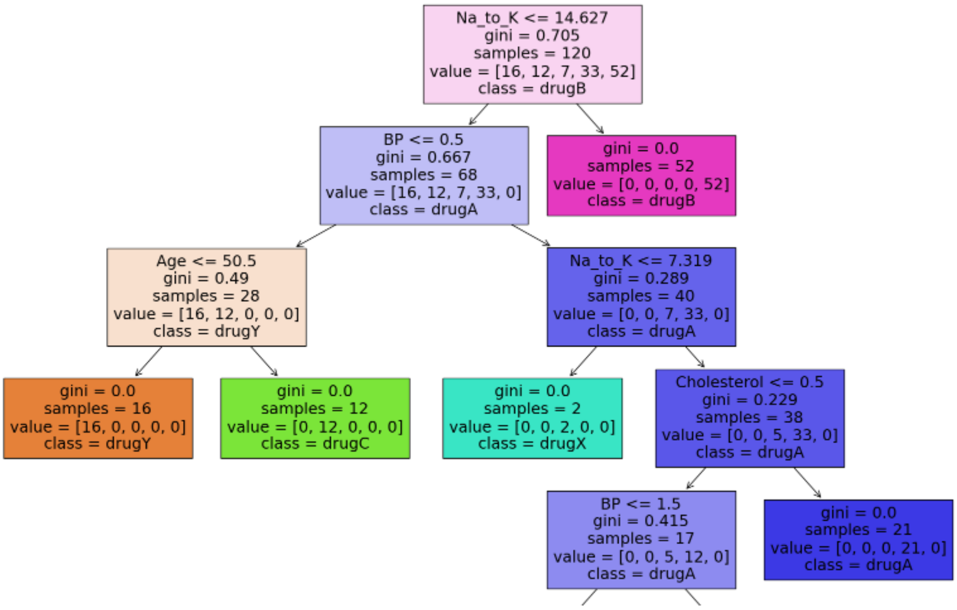
```
Classification Report:
               precision    recall  f1-score   support

       drugA       0.92      1.00      0.96        23
       drugB       1.00      0.88      0.93        16
       drugC       1.00      1.00      1.00        16
       drugX       1.00      0.98      0.99        54
       drugY       0.99      1.00      0.99        91

    accuracy                           0.98       200
   macro avg       0.98      0.97      0.98       200
weighted avg       0.99      0.98      0.98       200
```

```
sns.heatmap(confusion_matrix(y, predict), annot=True)
plt.show()
```



Based on the values, the Decision Tree algorithm graph was drawn.

```
cols = df.drop('Drug', axis=1).columns
classes = df['Drug'].unique()
plt.figure(figsize=(20,15))
tree.plot_tree(tree_model, feature_names=cols, class_names=classes,filled=True)
plt.show()
```



Hyperparameters`min_impurity_decrease' - defines how "clean" the result will be. The default value is 0

```
tree_model = DecisionTreeClassifier(min_impurity_decrease=0.04)
tree_model.fit(X_train, y_train)
y_predict = tree_model.predict(X_test)
print("DecisionTrees's Accuracy: ", accuracy_score(y_test, y_predict))
plt.figure(figsize=(20,15))
tree.plot_tree(tree_model, feature_names=cols, class_names=classes, filled=True)
plt.show()
```

```
DecisionTrees's Accuracy:  0.875
```

Using the max_dpth parameter, it is possible to control tree branches, that is, tree branch layers

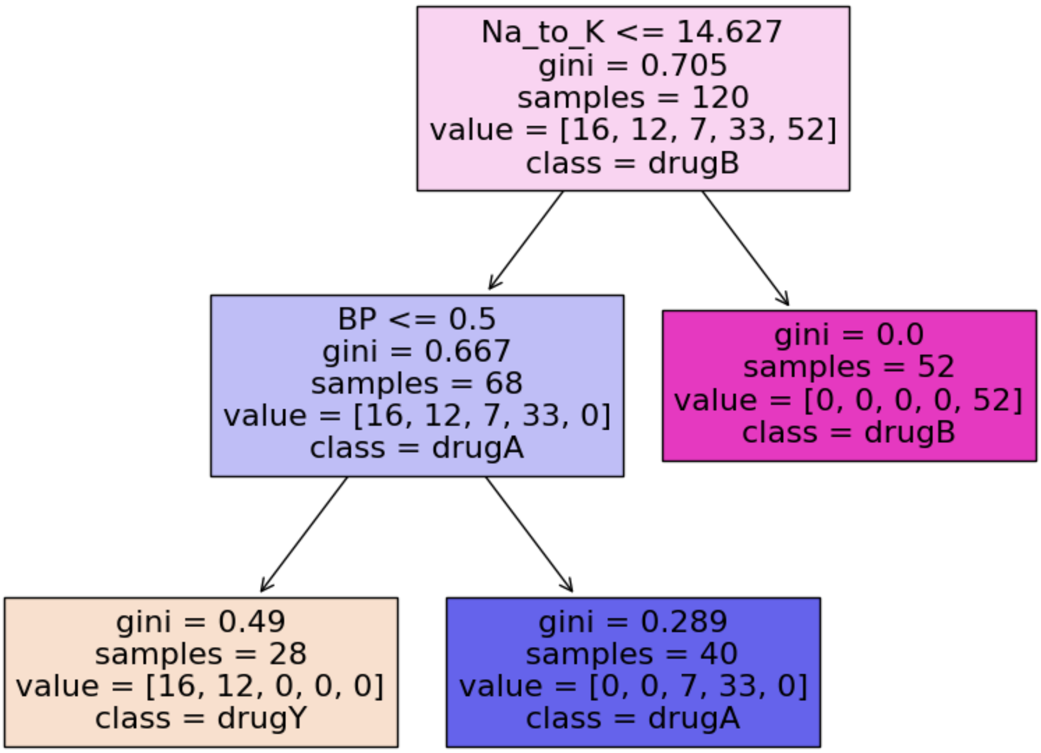
```
tree_model = DecisionTreeClassifier(min_impurity_decrease=0.02, max_depth=2)
tree_model.fit(X_train, y_train)
y_predict = tree_model.predict(X_test)
print("DecisionTrees's Accuracy: ", accuracy_score(y_test, y_predict))
plt.figure(figsize=(10,8))
tree.plot_tree(tree_model, feature_names=cols, class_names=classes, filled=True)
plt.show()
```

```
DecisionTrees's Accuracy:  0.825
```

min_samp_leaf The number of products to produce a leaf node (final leaf node).

```
tree_model = DecisionTreeClassifier(min_impurity_decrease=0.04,min_samples_leaf=10,max_depth=2)
tree_model.fit(X_train, y_train)
y_predict = tree_model.predict(X_test)
print("DecisionTrees's Accuracy: ", accuracy_score(y_test, y_predict))
plt.figure(figsize=(10,8))
tree.plot_tree(tree_model, feature_names=cols, class_names=classes, filled=True)
plt.show()
```

```
DecisionTrees's Accuracy:  0.825
```



Comparing graphs of the Decision Tree model, using hyperparameters significantly simplifies the model, and the effect on model accuracy is not significant.

References:

1. Amrullayevich K. A., Obid o'g'li S. J. ELEKTRON TALIM MUHITIDA TALABALARDA AXBOROT BILAN ISHLASH KOMPETENTLIKNI SHAKLLANTIRISH //International Journal of Contemporary Scientific and Technical Research. – 2022. – С. 641-645.

2. Obid o’g A. S. J. et al. Numpy Library Capabilities. Vectorized Calculation In Numpy Va Type Of Information //Eurasian Research Bulletin. – 2022. – Т. 15. – С. 132-137.

3. Javlon X. et al. Классификатор движения рук с использованием биомиметического распознавания образов с помощью сверточных нейронных сетей с методом динамического порога для извлечения движения с использованием датчиков EF //Journal of new century innovations. – 2022. – Т. 19. – №. 6. – С. 352-357.

4. Фитратович В. и др. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ МНОГОФАЗНОЙ ФИЛЬТРАЦИИ В НЕФТЕГАЗОВОМ ПЛАСТЕ ПРИ ЕГО ЗАВОДНЕНИИ //INTERNATIONAL CONFERENCES ON LEARNING AND TEACHING. – 2022. – Т. 1. – №. 4. – С. 520-525.

5. Jamshid S. ENTROPY EVALUATION CRITERION IN DECISION TREE ALGORITHM EVALUATION //International Journal of Contemporary Scientific and Technical Research. – 2023. – С. 236-239.

FINGER PRINT-BASED ATTENDANCE SYSTEM

Sherbaev Javokhir Ravshan ugli,
Abdurakhmanov Ravshan Anarbayevich
Jizzakh branch of National University of Uzbekistan

Annotation: The Fingerprint-Based Attendance System has emerged as a robust and secure method for accurately recording attendance in various organizations and educational institutions. This research paper explores the development, implementation, and evaluation of such a system, highlighting its advantages, challenges, and potential future enhancements. Through a combination of literature review and practical experimentation, this paper aims to provide insights into the effectiveness and reliability of fingerprint-based attendance systems.

Keywords: Fingerprint, Attendance Management, Authentication.

Attendance tracking is a crucial aspect of organizational management and educational institutions. Traditional methods of taking attendance, such as manual paper-based systems or card swiping, have proven to be inefficient and susceptible to fraud. In contrast, fingerprint-based attendance systems offer a more secure, accurate, and convenient solution.