Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

РК №2

По курсу
«Технологии машинного обучения»

Подготовил:

Студент группы

ИУ5-63Б Борисов А.М.

01.06.2022

Проверил:

2022 г.

Задание

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Вариант

ИУ5-63Б, ИУ5Ц-83Б	Дерево решений	Случайный лес

Датасет

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Решение:

```
[1] from sklearn.datasets import *
    import pandas as pd
    import numpy as np
    from sklearn import tree
    from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.preprocessing import StandardScaler
    from sklearn.ensemble import RandomForestRegressor
    from sklearn import metrics
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[2] data = pd.read_csv('heart.csv', sep=',')
```

Колонки датасета:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholestoral in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by flourosopy
- thal: 0 = normal; 1 = fixed defect; 2 = reversable defect
- The names and social security numbers of the patients were recently removed - - from the database, replaced with dummy values.

[3]
```
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
data.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

[ ]
```
data.shape
```

```
(1025, 14)
```

Проверим, содержатся ли пропуски в данных:

[ ]
```
#В нашем наборе нет пропусков
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[5]
```
#Разделение признаков
x = data.drop('target', axis=1)
y = data['target']
```

Дерево решений

[6]
```
#Построение модели
model=tree.DecisionTreeClassifier(criterion="entropy")
model.fit(x,y)
```

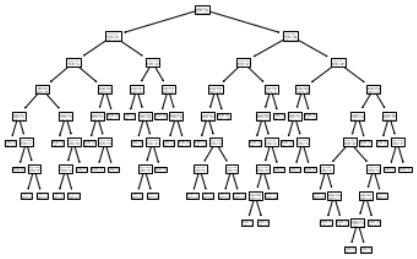
```
DecisionTreeClassifier(criterion='entropy')
```

[7]
```
#Оценка модели
model.score(x,y)
```

```
1.0
```

[8]
```
#Визуализирование данных
tree.plot_tree(model)
```

```
[Text(0.4855769230769231, 0.95, 'X[2] <= 0.5\nentropy = 0.999\nsamples = 1025\nvalue = [499, 526]'),
 Text(0.2692307692307692, 0.85, 'X[11] <= 0.5\nentropy = 0.804\nsamples = 497\nvalue = [375, 122]'),
 Text(0.17307692307692307, 0.75, 'X[12] <= 2.5\nentropy = 1.0\nsamples = 219\nvalue = [112, 107]'),
 Text(0.09615384615384616, 0.65, 'X[8] <= 0.5\nentropy = 0.833\nsamples = 125\nvalue = [33, 92]'),
 Text(0.038461538461538464, 0.55, 'X[7] <= 96.5\nentropy = 0.473\nsamples = 79\nvalue = [8, 71]'),
 Text(0.019230769230769232, 0.45, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(0.057692307692307696, 0.45, 'X[4] <= 316.5\nentropy = 0.3\nsamples = 75\nvalue = [4, 71]'),
 Text(0.038461538461538464, 0.35, 'entropy = 0.0\nsamples = 68\nvalue = [0, 68]'),
 Text(0.07692307692307693, 0.35, 'X[4] <= 362.0\nentropy = 0.985\nsamples = 7\nvalue = [4, 3]'),
 Text(0.057692307692307696, 0.25, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(0.09615384615384616, 0.25, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.15384615384615385, 0.55, 'X[6] <= 0.5\nentropy = 0.995\nsamples = 46\nvalue = [25, 21]'),
 Text(0.8846153846153846, 0.05, 'entropy = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(0.9038461538461539, 0.15, 'entropy = 0.0\nsamples = 12\nvalue = [12, 0]'),
 Text(0.9230769230769231, 0.25, 'entropy = 0.0\nsamples = 7\nvalue = [0, 7]'),
 Text(0.8846153846153846, 0.45, 'entropy = 0.0\nsamples = 14\nvalue = [14, 0]'),
 Text(0.9423076923076923, 0.55, 'X[10] <= 0.5\nentropy = 0.536\nsamples = 49\nvalue = [43, 6]'),
 Text(0.9230769230769231, 0.45, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.9615384615384616, 0.45, 'X[3] <= 119.0\nentropy = 0.348\nsamples = 46\nvalue = [43, 3]'),
 Text(0.9423076923076923, 0.35, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.9807692307692307, 0.35, 'entropy = 0.0\nsamples = 43\nvalue = [43, 0]')]
```



[14]
```
#Разделим данные
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

[15]
```
regressor = DecisionTreeRegressor()
regressor.fit(x_train, y_train)
```

```
DecisionTreeRegressor()
```

[16]
```
y_pred = regressor.predict(x_test)
print('Дерево решений')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Дерево решений
Mean Absolute Error: 0.00974025974025974
Mean Squared Error: 0.00974025974025974
Root Mean Squared Error: 0.09869275424396534
```

[21]
```
#Доля правильных ответов в обучающей и тестовой выборках
print('Дерево решений')
print("Accuracy on training set: {:.3f}".format(regressor.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(regressor.score(x_test, y_test)))
```

```
Дерево решений
Accuracy on training set: 1.000
Accuracy on test set: 0.961
```

Случайный лес

```
[22] regressor = RandomForestRegressor(n_estimators=20, random_state=0)
     regressor.fit(x_train, y_train)
     y_pred = regressor.predict(x_test)
```

```
[23] print('Случайный лес')
     print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
     print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Случайный лес
Mean Absolute Error: 0.048863636363636366
Mean Squared Error: 0.008758116883116885
Root Mean Squared Error: 0.09358481117743886
```

```
[24] #Доля правильных ответов в обучающей и тестовой выборках
     print('Случайный лес')
     print("Accuracy on training set: {:.3f}".format(regressor.score(x_train, y_train)))
     print("Accuracy on test set: {:.3f}".format(regressor.score(x_test, y_test)))
```

```
Случайный лес
Accuracy on training set: 0.990
Accuracy on test set: 0.965
```

Анализ результатов:

Дерево решений

```
Дерево решений
Mean Absolute Error: 0.00974025974025974
Mean Squared Error: 0.00974025974025974
Root Mean Squared Error: 0.09869275424396534
```

```
Дерево решений
Accuracy on training set: 1.000
Accuracy on test set: 0.961
```

Случайный лес

```
Случайный лес
Mean Absolute Error: 0.048863636363636366
Mean Squared Error: 0.008758116883116885
Root Mean Squared Error: 0.09358481117743886
```

```
Случайный лес
Accuracy on training set: 0.990
Accuracy on test set: 0.965
```

Точность моделей дерева решений и случайного леса практически одинакова.