Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчет по ЛР №6

по курсу «Технологии машинного обучения»

«Ансамбли моделей машинного обучения»

ИСПОЛНИТЕЛЬ:
Аушева Л.И.
Группа ИУ5-61Б

_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

_____

"__"_____2020 г.

Москва 2020

Цель лабораторной работы:

Изучение ансамблей моделей машинного обучения.

Задание:

1. Выберите набор данных (diabetes.csv) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Выполнение:

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score, balanced_accuracy_sco
re
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier

%matplotlib inline
sns.set(style="ticks")
```

In [2]:
```
filepath = "diabetes.csv"

data = pd.read_csv(filepath, sep=',')
data
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [4]:
```
data['Outcome'].value_counts()
```

Out[4]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [5]:
```
# посчитаем дисбаланс классов
total = data.shape[0]
class_0, class_1 = data['Outcome'].value_counts()
print('Класс 0 составляет {}%, а класс 1 составляет {}%.'
      .format(round(class_0 / total, 2)*100, round(class_1 / total, 2)*100))
```

```
Класс 0 составляет 65.0%, а класс 1 составляет 35.0%.
```

In [6]:
```
# Разделим данные на целевой столбец и признаки
X = data.drop("Outcome", axis=1)
Y = data["Outcome"]
print(X, "\n")
print(Y)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]

0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [7]:
```
# Предобработаем данные, чтобы методы работали лучше
columns = X.columns
scaler = StandardScaler()
X = scaler.fit_transform(X)
pd.DataFrame(X, columns=columns).describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
count	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02
mean	-6.476301e-17	-9.251859e-18	1.503427e-17	1.006140e-16	-3.006854e-17	2.590520e-16	2.451743e-16
std	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00
min	-1.141852e+00	-3.783654e+00	-3.572597e+00	-1.288212e+00	-6.928906e-01	-4.060474e+00	-1.189553e+00
25%	-8.448851e-01	-6.852363e-01	-3.673367e-01	-1.288212e+00	-6.928906e-01	-5.955785e-01	-6.889685e-01
50%	-2.509521e-01	-1.218877e-01	1.496408e-01	1.545332e-01	-4.280622e-01	9.419788e-04	-3.001282e-01
75%	6.399473e-01	6.057709e-01	5.632228e-01	7.190857e-01	4.120079e-01	5.847705e-01	4.662269e-01
max	3.906578e+00	2.444478e+00	2.734528e+00	4.921866e+00	6.652839e+00	4.455807e+00	5.883565e+00

In [8]:
```
# С использованием метода train_test_split разделим выборку на обучающую и тестовую
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("Y_train:", Y_train.shape)
print("Y_test:", Y_test.shape)
```

```
X_train: (576, 8)
X_test: (192, 8)
Y_train: (576,)
Y_test: (192,)
```

In [9]:
```
def test_model(model):
    print("balanced_accuracy_score:",
          balanced_accuracy_score(Y_test, model.predict(X_test)))
    print("f1_score:",
          f1_score(Y_test, model.predict(X_test)))
```

Случайный лес

In [10]:
```
RF = RandomForestClassifier(n_estimators=20, oob_score=True, random_state=10)
RF.fit(X_train, Y_train)
```

Out[10]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=True, random_state=10, verbose=0,
                       warm_start=False)
```

In [11]:
```
test_model(RF)
```

```
balanced_accuracy_score: 0.7537999293036408
f1_score: 0.6821705426356589
```

Бэггинг

In [12]:
```
BC = BaggingClassifier(n_estimators=20, oob_score=True, random_state=10)
BC.fit(X_train, Y_train)
```

Out[12]:
```
BaggingClassifier(base_estimator=None, bootstrap=True, bootstrap_features=False,
                  max_features=1.0, max_samples=1.0, n_estimators=20,
                  n_jobs=None, oob_score=True, random_state=10, verbose=0,
                  warm_start=False)
```

```
In [13]: test_model(BC)
```

```
balanced_accuracy_score: 0.707140332272888
f1_score: 0.6115702479338844
```

Вывод:

Сравнивая качество моделей, можно сделать вывод, что случайный лес показал лучшие практические результаты.