МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э.Баумана

Отчет по лабораторной работе №5
по курсу «Технологии машинного обучения»

Ансамбли моделей машинного обучения.

Подготовил
Ионов С.А.
ИУ5-62Б

1) Описание задания

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
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.

2) Текст программы

0) Подготовка

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, validation_curve, learning_curve
from sklearn.metrics import confusion_matrix, classification_report, plot_roc_curve, f1_score

from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier

data = pd.read_csv('data/data.csv', sep=',')

# Проверка дубликатов среди ID
data['id'].value_counts().count() - data.shape[0]

# Удаляем незначащие столбцы
data.drop(columns=['id', 'Unnamed: 32'], inplace=True)

# Проверяем наличие пропусков
data.isnull().sum()[data.isnull().sum() != 0]

# Проверяем баланс классов целевой переменной
pd.concat([data['diagnosis'].value_counts(),
round(data['diagnosis'].value_counts()/data.shape[0]*100, 1)],
        keys=['Количество', 'Процент'], axis=1)

# Кодируем целевую переменную
data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})
```

```
plt.figure(figsize=(20,20))
g = sns.heatmap(data.corr(), annot=True, fmt='.2f')

data = data.drop(columns=['fractal_dimension_mean', 'texture_se', 'symmetry_se'])

TEST_SIZE = 0.3
RANDOM_STATE = 0

data_X = data.drop(columns='diagnosis')
data_y = data['diagnosis']
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data_X, data_y,
test_size=TEST_SIZE, \
                                                        random_state=RANDOM_STATE)

def print_metrics(X_train, Y_train, X_test, Y_test, clf):
    clf.fit(X_train, Y_train)
    target = clf.predict(X_test)
    ret = f1_score(Y_test, target)
    print(f'F1-мера: {ret}')
    fig, ax = plt.subplots()
    plot_roc_curve(clf, X_test, Y_test, ax=ax)
    ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
            label='Chance', alpha=.8)
    ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
           title="Receiver operating characteristic")
    ax.legend(loc="lower right")
    plt.show()
    print(f'Матрица ошибок:\n {confusion_matrix(Y_test, target)}')
    print(classification_report(Y_test, target, target_names=['B', 'M']))
    return ret

def plot_learning_curve(data_X, data_y, clf):
    train_sizes, train_scores, test_scores = learning_curve(estimator=clf, scoring='f1', X=data_X,
y=data_y,
                                                            train_sizes=np.linspace(0.1, 1.0, 10), cv=5)
    train_mean = np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)
    plt.figure(figsize=(7,5))
    plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5,
label=f'тренировочная f1-мера')
    plt.fill_between(train_sizes, train_mean+train_std, train_mean-train_std, alpha=0.15,
color='blue')
    plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize=5,
             label=f'проверочная f1-мера')
    plt.fill_between(train_sizes, test_mean+test_std, test_mean-test_std, alpha=0.15,
color='green')
```

```
    plt.grid()
    plt.legend(loc='lower right')
    plt.xlabel('Число тренировочных образцов')
    plt.ylabel('f1-мера')
    plt.show()
```

1) Случайный лес

```
rfc = RandomForestClassifier(random_state=RANDOM_STATE)
res_rfc = print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, rfc)

plot_learning_curve(data_X, data_y, rfc)

def show_feature_importance(importance, col_names):
    data =
pd.DataFrame({'feature_names':np.array(col_names),'feature_importance':np.array(importance
)})
    data.sort_values(by=['feature_importance'], ascending=False,inplace=True)

    plt.figure(figsize=(10,7))
    sns.barplot(x=data['feature_importance'], y=data['feature_names'])
    plt.title('Feature importance using DecisionTreeClassifier')
    plt.xlabel('importance')
    plt.ylabel('name')

show_feature_importance(rfc.feature_importances_, data_X.columns.tolist())
```

2) SVM

```
boost_model = CatBoostClassifier(eval_metric='F1', random_seed=RANDOM_STATE)
res_boost = print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, boost_model)

plot_learning_curve(data_X, data_y, boost_model)
```

3) Итоги

```
lst_label = ['Random forest', 'Cat boost']
dc_score = {'F1-measure':[res_rfc, res_boost]}
pd.DataFrame(dc_score, index=lst_label)
```

3) Экранные формы с примерами выполнения программы

1. Подготовка

- Для лабораторной работы был выбран датасет [Breast Cancer Wisconsin](). Будем решать задачу классификации для предсказания наличия рака молочной железы.

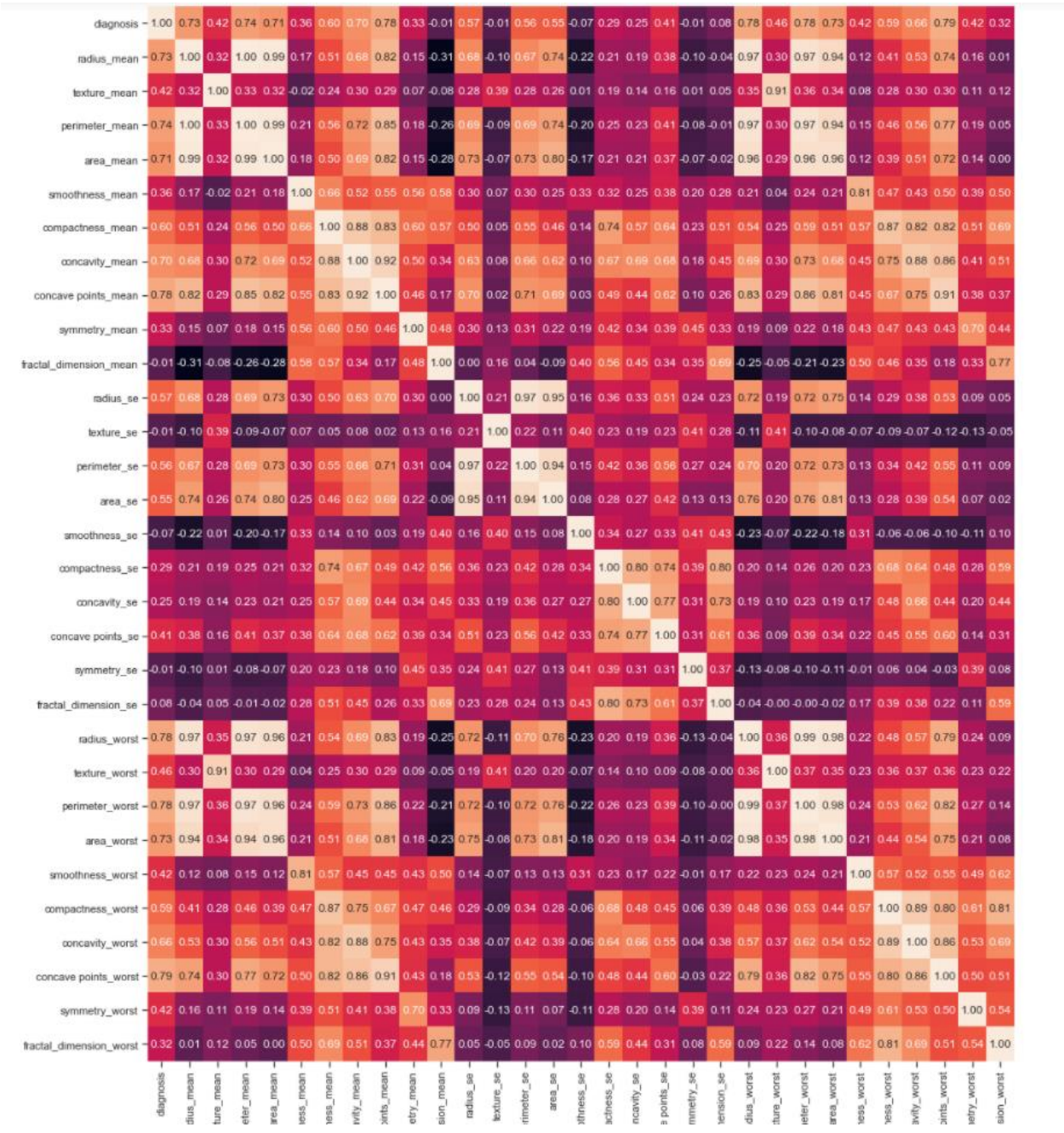
- Состав признаков в датасете:

```
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
```

- Баланс классов допустим:

	Количество	Процент
B	357	62.7
M	212	37.3

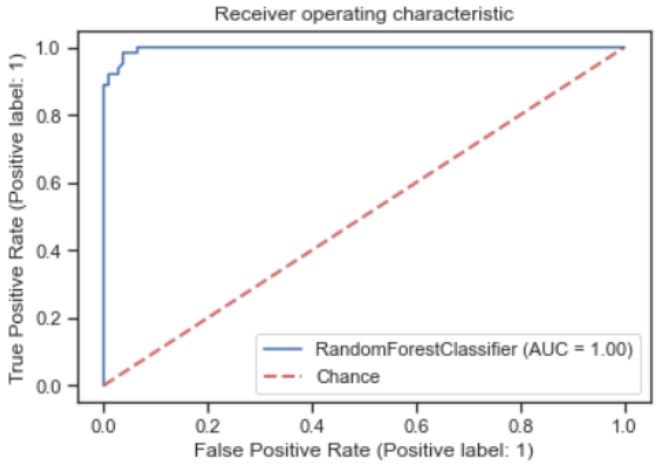
- Из анализа корреляционной матрицы принимается решение об удалении наименее слабо коррелируемых признаков с целевым (коэффициент корреляции равен 0,01 по модулю): "fractal_dimension_mean", "texture_se", а также "symmetry_se".



2. Случайный лес

- Получаем следующую оценку:

```
F1-мера: 0.9612403100775193
```

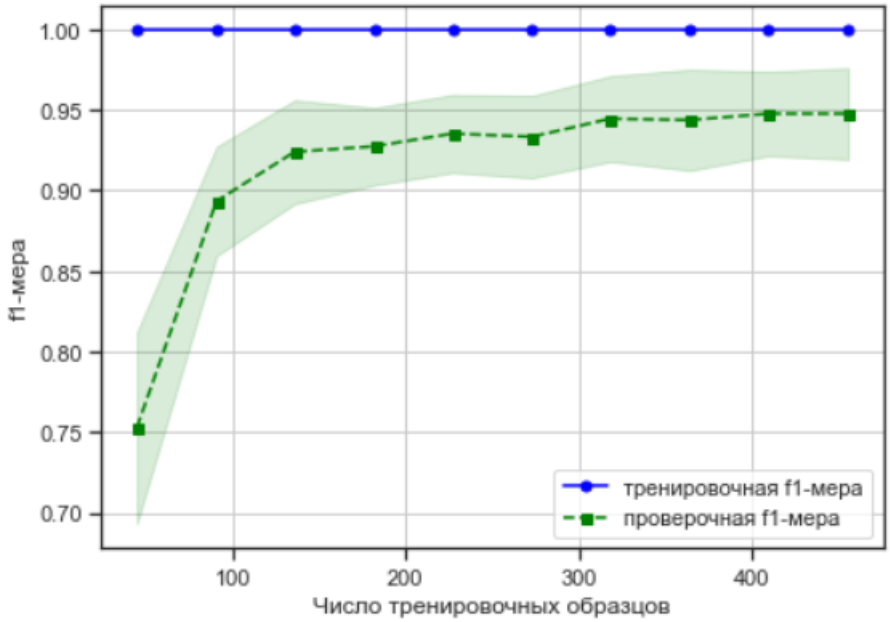



```
Матрица ошибок:
 [[104   4]
 [  1  62]]
              precision    recall  f1-score   support

           B       0.99      0.96      0.98       108
           M       0.94      0.98      0.96        63

    accuracy                           0.97       171
   macro avg       0.96      0.97      0.97       171
weighted avg       0.97      0.97      0.97       171
```

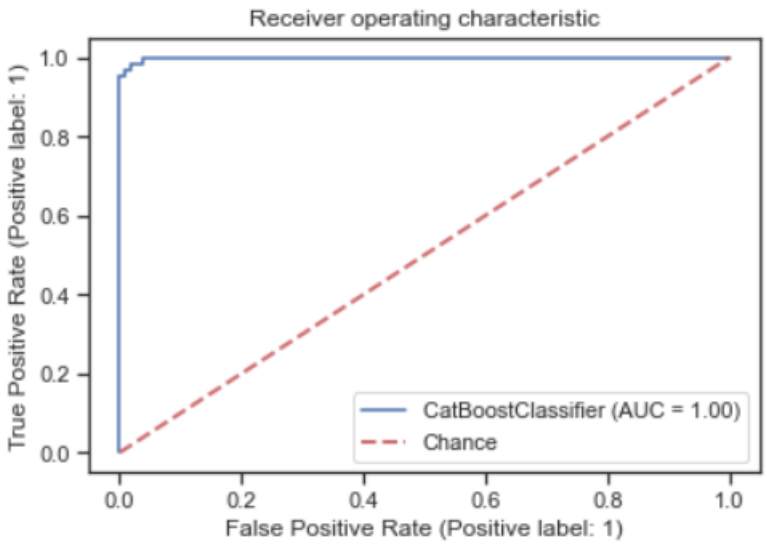
- Кривая обучения для модели:



3. Бустинг

- Получаем следующую оценку:

```
F1-мера: 0.9763779527559054
```

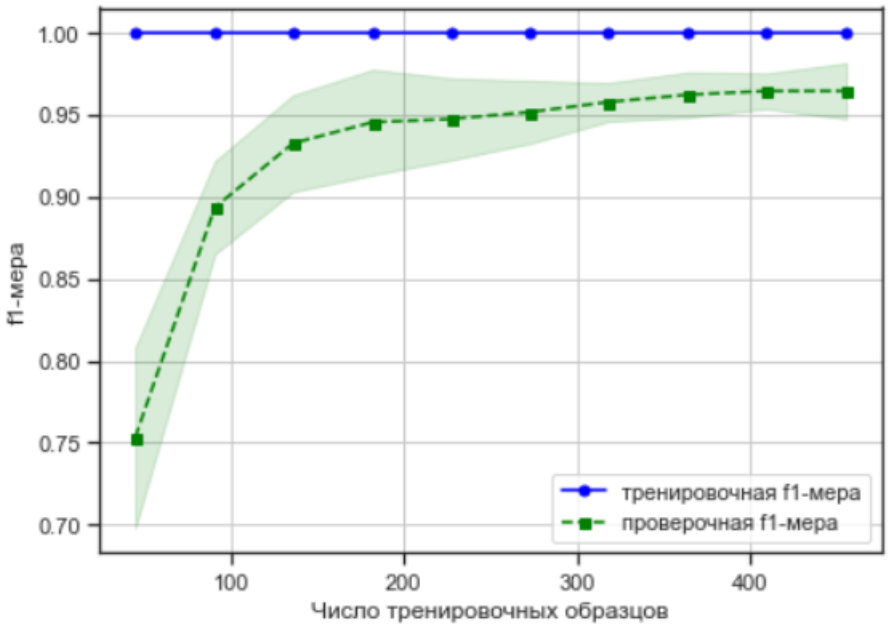



```
Матрица ошибок:
 [[106   2]
 [  1  62]]
              precision    recall  f1-score   support

           B       0.99      0.98      0.99       108
           M       0.97      0.98      0.98        63

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171
```

- Кривая обучения для модели:



4. Итоги

- В результате сравнения результатов анализа по метрике «f1_score», позволяющей оценивать баланс между «precision» (отвечает за минимизацию ситуаций, когда здорового приняли за больного) и «recall» (отвечает за минимизацию ситуаций, когда больного приняли за здорового), можно сделать вывод о том, что в данных есть сложные взаимосвязи, поэтому бустинг позволяет их обнаруживать более точно, чем случайный лес.

	F1-measure
Random forest	0.961240
Cat boost	0.976378