Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

«Методы машинного обучения»

Отчет по Рубежному контролю №2

Выполнила:

студентка группы ИУ5-22М

Петрова Ирина

Проверил:

доцент, к.т.н.

Гапанюк Ю. Е.

Москва, 2020

Рубежный контроль №2

Петрова Ирина ИУ5-22М

Вариант 11

Задача №2. ¶

Для заданного набора данных постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения модели используйте ансамблевые модели: случайный лес и градиентный бустинг. Оцените качество модели на основе подходящих метрик качества (не менее трех метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей?

Набор данных: https://www.kaggle.com/karangadiya/fifa19 (https://www.kaggle.com/karangadiya/fifa19)

Решается **задача регрессии** с использованием ключевого признака "Overall"

In [1]:

```
import numpy as np import
pandas as pd import seaborn as
sns import matplotlib.pyplot as
plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_rep
ort
from sklearn.metrics import confusion_matrix from
sklearn.metrics import plot_confusion_matrix from
sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_e
rror, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR from
sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor from
sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor from
sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor from
gmdhpy import gmdh %matplotlib inline  sns.set(style="ticks")
```

In [2]:

```
data = pd.read_csv('D:/Загрузки/fifa19/data.csv', sep=",")
data.head()
```

Out[2]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	ht

1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	ht	
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	ht	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	ht	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	h	
5	rows × 89 columns							

In [3]:

```
# размер набора данных
data.shape
```

Out[3]:

```
(18207, 89)
```

In [4]:

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 18207
```

Обработка пропусков в данных

In [5]:

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[5]:

```
Unnamed: 0            0
ID                    0
Name                  0
Age                   0
Photo                 0
...
GKHandling           48
GKKicking            48
GKPositioning        48
GKReflexes           48
Release Clause     1564

Length: 89, dtype: int64
```

In [6]:

```
# Удаление колонок, содержащих пустые значения
data_all = data.dropna(axis=1, how='any')
(data.shape, data_all.shape)
```

Out[6]: ((18207, 89),

(18207, 13)) In [7]:

```
data_all.head()
```

Out[7]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	ht
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	ht
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	ht
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	ht
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	h

Выбор признаков, подходящих для построения моделей.

In [8]:

```
data_all.dtypes
```

Out[8]:

```
Unnamed: 0      int64
ID              int64
Name           object
Age             int64
Photo          object
Nationality    object
Flag           object
Overall         int64
Potential       int64
Club Logo      object
Value          object
Wage           object
Special         int64 dtype:
object
```

Для построения моделей будем использовать только количественные признаки, кроме признака "Unnamed: 0".

Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения

In [9]:

```
corr_cols = ['ID', 'Age', 'Overall', 'Potential', 'Special']
corr_cols
```

Out[9]:

```
['ID', 'Age', 'Overall', 'Potential', 'Special']
```

In [10]:

```
fig, ax = plt.subplots(figsize=(10,5))
sns.heatmap(data_all[corr_cols].corr(), annot=True, fmt='.2f')
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84383c240>
```

На основе корреляционной матрицы можно сделать следующие выводы:

- Целевой признак регрессии "Overall" наиболее сильно коррелирует с "Potential" (0.66) и "Special" (0.61). Эти признаки обязательно следует оставить в модели регрессии.
- Большие по модулю значения коэффициентов корреляции свидетельствуют о значимой корреляции между исходными признаками и целевым признаком. На основании корреляционной

матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

Выбор метрик для последующей оценки качества моделей.

В качестве метрик для решения задачи регрессии будем использовать:

1. Mean absolute error - средняя абсолютная ошибка

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Чем ближе значение к нулю, тем лучше качество регрессии.

Основная проблема метрики состоит в том, что она не нормирована.

Вычисляется с помощью функции mean_absolute_error.

1. Mean squared error - средняя квадратичная ошибка

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Вычисляется с помощью функции mean_squared_error.

1. Метрика R2 или коэффициент детерминации

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Вычисляется с помощью функции r2_score.

In [11]:

```
class MetricLogger:
        def
 __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
 'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
 Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index,
inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
 self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
 return temp_data_2['alg'].values, temp_data_2['value'].values
        def plot(self, str_header, metric, ascending=True, figsize=(5,
 5)):         """
        Вывод графика
 """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
 fig, ax1 = plt.subplots(figsize=figsize)        pos =
 np.arange(len(array_metric))        rects = ax1.barh(pos, array_metric,
 align='center',                         height=0.5,
 tick_label=array_labels)        ax1.set_title(str_header)        for a,b in
 zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
 plt.show()
```

Формирование обучающей и тестовой выборок на основе исходного набора данных

In [12]:

```
# Признаки для задачи регресии
task_regr_cols = ['Potential', 'Special', 'Age']
```

In [13]:

```
# Разделение выборки на обучающую и тестовую
regr_X_train, regr_X_test, regr_Y_train, regr_Y_test = train_test_split(
    data_all[task_regr_cols], data_all['Overall'], test_size=0.5, random_state=1)
regr_X_train.shape, regr_X_test.shape, regr_Y_train.shape, regr_Y_test.shape
```

Out[13]:

```
((9103, 3), (9104, 3), (9103,), (9104,))
```

Решение задачи регрессии

In [14]:

```
# Модели
regr_models = {'RF':RandomForestRegressor(),
               'GB':GradientBoostingRegressor()}
```

In [15]:

```
# Сохранение метрик
regrMetricLogger = MetricLogger()
```

In [16]:

```
def regr_train_model(model_name, model, regrMetricLogger):
 model.fit(regr_X_train, regr_Y_train)
    Y_pred = model.predict(regr_X_test)

    mae = mean_absolute_error(regr_Y_test, Y_pred)
 mse = mean_squared_error(regr_Y_test, Y_pred)    r2
 = r2_score(regr_Y_test, Y_pred)

    regrMetricLogger.add('MAE', model_name, mae)
 regrMetricLogger.add('MSE', model_name, mse)
 regrMetricLogger.add('R2', model_name, r2)

    print('*******************************************************')
 print(model)    print()
    print('MAE={}, MSE={}, R2={}'.format(        round(mae, 3),
 round(mse, 3), round(r2, 3)))
 print('*******************************************************')
```

In [17]:

```
for model_name, model in regr_models.items():
 regr_train_model(model_name, model, regrMetricLogger)
```

```
**************************************************
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
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=100, n_jobs=None, oob_score=False,
random_state=None, verbose=0, warm_start=False)
MAE=1.035, MSE=2.516, R2=0.947
**************************************************
**************************************************
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_ms
e',
                          init=None, learning_rate=0.1, loss='ls', max_dep
th=3,
                          max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=No ne,
                          min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
n_iter_no_change=None, presort='deprecated',
random_state=None, subsample=1.0, tol=0.0001,
validation_fraction=0.1, verbose=0, warm_start=F alse)
```

```
MAE=1.006, MSE=2.083, R2=0.956
**************************************************
```

Формирование выводов о качестве построенных моделей на основе выбранных метрик.

In [18]:

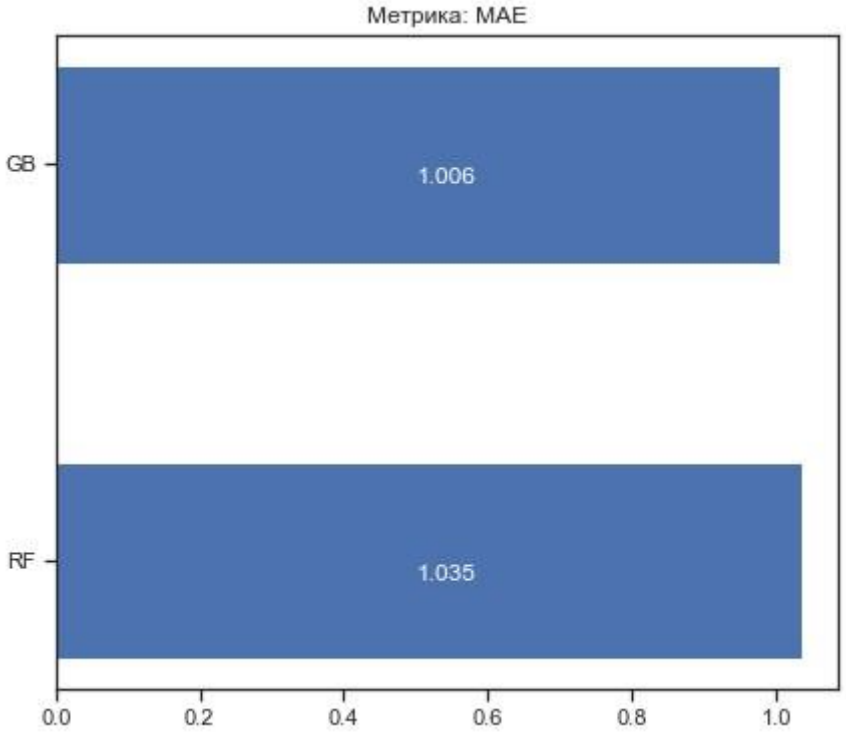
```
# Метрики качества модели
regr_metrics = regrMetricLogger.df['metric'].unique()
regr_metrics
```

Out[18]:

```
array(['MAE', 'MSE', 'R2'], dtype=object)
```

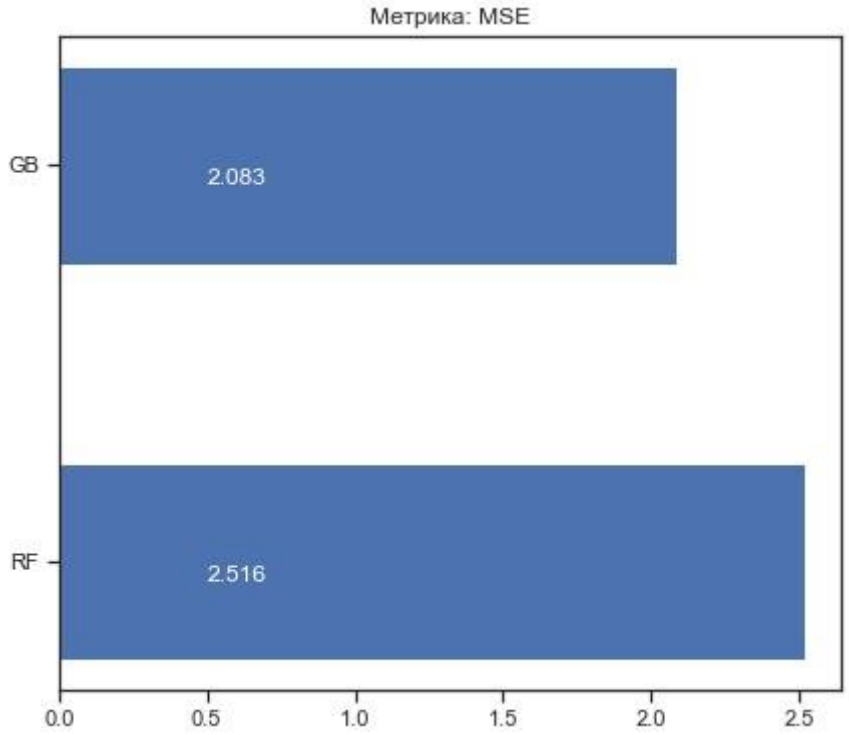
In [19]:

```
regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=(7, 6))
```



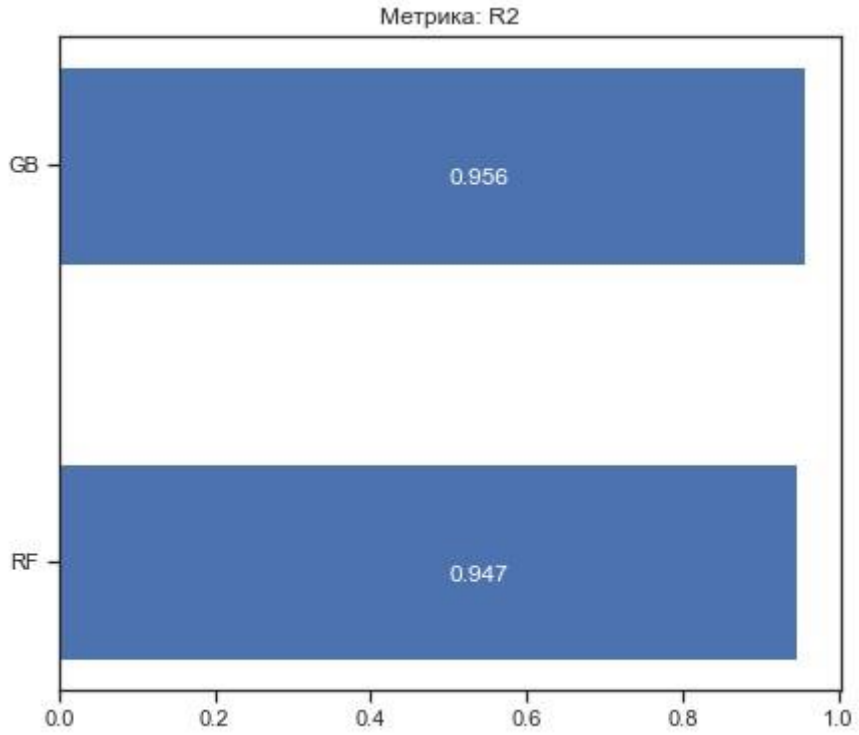
In [20]:

```
regrMetricLogger.plot('Метрика: ' + 'MSE', 'MSE', ascending=False, figsize=(7, 6))
```



In [21]:

```
regrMetricLogger.plot('Метрика: ' + 'R2', 'R2', ascending=True, figsize=(7, 6))
```



Вывод: лучшей оказалась модель на основе градиентного бустинга.