Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №1

«Технологии разведочного анализа и обработки данных.»

Вариант № 17

Выполнил
Сайфутдинов Р.И.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 15.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Задание:

Номер варианта: **17**

https://www.kaggle.com/mathan/fifa-2018-match-statistics

Группа	Метод №1	Метод №2
ИУ5-62Б, ИУ5Ц-82Б	Метод опорных векторов	Случайный лес

Задача №3.

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.

Ход выполнения:

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.

Условия по варианту (ИУ5-62Б Вар.17) ИУ5-62Б, ИУ5Ц-82Б Метод опорных векторов Случайный лес

17. https://www.kaggle.com/mathan/fifa-2018-match-statistics

Загрузка датасета

```
[24] import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler, LabelEncoder
     from sklearn.svm import SVC
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

     # Загрузка данных
     df = pd.read_csv('FIFA 2018 Statistics.csv')
     print("Первые 5 строк данных:\n", df.head())
     print("\nИнформация о данных:\n", df.info())
     print("\nПропуски в данных:\n", df.isnull().sum())
```

```
Первые 5 строк данных:
         Date          Team      Opponent  Goal Scored  Ball Possession %  \
0  14-06-2018        Russia  Saudi Arabia            5                 40
1  14-06-2018  Saudi Arabia        Russia            0                 60
2  15-06-2018         Egypt       Uruguay            0                 43
3  15-06-2018       Uruguay         Egypt            1                 57
4  15-06-2018       Morocco          Iran            0                 64

   Attempts  On-Target  Off-Target  Blocked  Corners  ...  Yellow Card  \
0        13          7           3        3        6  ...            0
1         6          0           3        3        2  ...            0
2         8          3           3        2        0  ...            2
3        14          4           6        4        5  ...            0
4        13          3           6        4        5  ...            1

   Yellow & Red  Red  Man of the Match  1st Goal        Round  PSO  \
0             0    0               Yes      12.0  Group Stage   No
1             0    0                No       NaN  Group Stage   No
2             0    0                No       NaN  Group Stage   No
3             0    0               Yes      89.0  Group Stage   No
4             0    0                No       NaN  Group Stage   No

   Goals in PSO  Own goals  Own goal Time
0             0        NaN            NaN
1             0        NaN            NaN
2             0        NaN            NaN
3             0        NaN            NaN
4             0        1.0           90.0

[5 rows x 27 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 27 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Date                     128 non-null    object
 1   Team                     128 non-null    object
 2   Opponent                 128 non-null    object
 3   Goal Scored              128 non-null    int64
 4   Ball Possession %        128 non-null    int64
 5   Attempts                 128 non-null    int64
 6   On-Target                128 non-null    int64
 7   Off-Target               128 non-null    int64
 8   Blocked                  128 non-null    int64
 9   Corners                  128 non-null    int64
 10  Offsides                 128 non-null    int64
 11  Free Kicks               128 non-null    int64
 12  Saves                    128 non-null    int64
 13  Pass Accuracy %          128 non-null    int64
 14  Passes                   128 non-null    int64
 15  Distance Covered (Kms)   128 non-null    int64
 16  Fouls Committed          128 non-null    int64
 17  Yellow Card              128 non-null    int64
 18  Yellow & Red             128 non-null    int64
 19  Red                      128 non-null    int64
 20  Man of the Match         128 non-null    object
 21  1st Goal                 94 non-null     float64
 22  Round                    128 non-null    object
 23  PSO                      128 non-null    object
 24  Goals in PSO             128 non-null    int64
 25  Own goals                12 non-null     float64
 26  Own goal Time            12 non-null     float64
dtypes: float64(3), int64(18), object(6)
memory usage: 27.1+ KB

Информация о данных:
 None

Пропуски в данных:
 Date                       0
Team                        0
Opponent                    0
Goal Scored                 0
Ball Possession %           0
Attempts                    0
On-Target                   0
Off-Target                  0
Blocked                     0
Corners                     0
Offsides                    0
Free Kicks                  0
Saves                       0
Pass Accuracy %             0
Passes                      0
Distance Covered (Kms)      0
Fouls Committed             0
Yellow Card                 0
Yellow & Red                0
Red                         0
Man of the Match            0
1st Goal                   34
Round                       0
PSO                         0
```

Обработка пропусков

```
[25] # Заполнение пропусков
     df.fillna({
         'Own goals': 0,
         'Own goal Time': 0,
         '1st Goal': df['1st Goal'].median(),
         'Corners': df['Corners'].median()
     }, inplace=True)
```

Удаление или преобразование нечисловых столбцов

```
[26] # Удаляем столбцы, которые не могут быть преобразованы в числа (например, даты)
     df = df.drop(['Date'], axis=1)  # Удаляем столбец с датой

     # Label Encoding для категориальных признаков (если они остались)
     categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
     for col in categorical_cols:
         if col != 'Man of the Match':  # Целевая переменная
             df[col] = LabelEncoder().fit_transform(df[col])

     # One-Hot Encoding для целевой переменной (если нужно)
     df['Man of the Match'] = LabelEncoder().fit_transform(df['Man of the Match'])
```

Масштабирование числовых признаков

```
[27] numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
     numerical_cols.remove('Man of the Match')  # Исключаем целевую переменную

     scaler = StandardScaler()
     df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

Разделение данных на обучающую и тестовую выборки

```
[28] X = df.drop('Man of the Match', axis=1)
     y = df['Man of the Match']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Обучение моделей

```
[29] # SVM
     svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
     svm_model.fit(X_train, y_train)
     y_pred_svm = svm_model.predict(X_test)

     # Random Forest
     rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
     rf_model.fit(X_train, y_train)
     y_pred_rf = rf_model.predict(X_test)
```

Оценка качества моделей

```
[30] # Метрики для SVM
     print("\nSVM Metrics:")
     print("Accuracy:", accuracy_score(y_test, y_pred_svm))
     print("Precision:", precision_score(y_test, y_pred_svm))
     print("Recall:", recall_score(y_test, y_pred_svm))
     print("F1-Score:", f1_score(y_test, y_pred_svm))

     # Метрики для Random Forest
     print("\nRandom Forest Metrics:")
     print("Accuracy:", accuracy_score(y_test, y_pred_rf))
     print("Precision:", precision_score(y_test, y_pred_rf))
     print("Recall:", recall_score(y_test, y_pred_rf))
     print("F1-Score:", f1_score(y_test, y_pred_rf))
```

```
SVM Metrics:
Accuracy: 0.5128205128205128
Precision: 0.6666666666666666
Recall: 0.4166666666666667
F1-Score: 0.5128205128205128

Random Forest Metrics:
Accuracy: 0.6666666666666666
Precision: 0.9230769230769231
Recall: 0.5
F1-Score: 0.6486486486486487
```

Какие метрики использовались и почему?

Accuracy — общая точность классификации.

Precision — доля верно предсказанных `Man of the Match` среди всех предсказанных.

Recall — доль верно предсказанных `Man of the Match` среди всех реальных.

F1-Score — гармоническое среднее Precision и Recall (важно при дисбалансе классов).

Сравнение моделей

1. Общая эффективность моделей

Метрика	SVM	Random Forest
Accuracy	0.51 (51%)	**0.67 (67%)**
Precision	0.67 (67%)	**0.92 (92%)**
Recall	0.42 (42%)	0.50 (50%)
F1-Score	0.51 (51%)	**0.65 (65%)**

Random Forest показал лучшие результаты по всем метрикам.

SVM хуже справляется с дисбалансом классов (меньше Recall).

Random Forest лучше обобщает данные благодаря ансамблевому подходу.