



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«МИРЭА – Российский технологический университет»**

**РГУ МИРЭА**

---

---

**Институт информационных технологий (ИИТ)  
Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  
по дисциплине «Технологии и инструментарий анализа больших данных»

**Практическая работа № 5**

Студент группы

*ИКБО-01-22 Прокопчук Роман Олегович*

---

(подпись)

Ассистент

*Тетерин Николай Николаевич*

---

(подпись)

Отчёт представлен

*«\_\_» ноября 2025 г.*

Москва, 2025 г.

## **ЦЕЛЬ**

Ознакомиться с инструментами классификации данных на Python.

## **ХОД РАБОТЫ**

### **Задание**

1. Найти данные для классификации. Данные в группе повторяться не должны. Предобработать данные, если это необходимо.
2. Изобразить гистограмму, которая показывает баланс классов. Сделать выводы.
3. Разбить выборку на тренировочную и тестовую. Тренировочная для обучения модели, тестовая для проверки ее качества.
4. Применить алгоритмы классификации: логистическая регрессия, SVM, KNN. Построить матрицу ошибок по результатам работы моделей (использовать `confusion_matrix` из `sklearn.metrics`). 5. Сравнить результаты классификации, используя `accuracy`, `precision`, `recall` и `f1`-меру (можно использовать `classification_report` из `sklearn.metrics`). Сделать выводы.
6. Оформить отчет о проделанной работе.

### **Ход работы**

В качестве данных был выбран датасет с результатами гонок Формулы 1 – <https://www.kaggle.com/datasets/ignale/formula-one-races-results-1950-2022>.

Код для выполнения задания 1 представлен на рисунке 1. Результат его работы – на рисунке 2.

```

import pandas as pd

df = pd.read_csv(filepath_or_buffer='data/f1_data.csv', low_memory=False)

print("\nИсходные данные (первые строки):")
print(df.head())
print(f"\nВсего записей: {len(df)}")
print("\nСтолбцы в исходном датасете:", df.columns.tolist())

rename_map = {
    'Position Order': 'position_order',
    'Grid': 'grid',
    'Laps': 'laps',
    'Team': 'team',
}

existing_rename_map = {
    old: new for old, new in rename_map.items() if old in df.columns
}

df = df[list(existing_rename_map.keys())].rename(columns=existing_rename_map)

print("\nПосле выбора нужных столбцов:")
print(df.head())
print("\nСтолбцы после переименования:", df.columns.tolist())

for col in ['position_order', 'grid', 'laps']:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')

df = df.dropna(subset=['position_order'])

if 'grid' in df.columns:
    df = df[df['grid'] > 0]
if 'laps' in df.columns:
    df = df[df['laps'] > 0]

feature_cols = ['grid', 'laps']

if 'team' in df.columns:
    df['team'] = df['team'].fillna('Unknown')
    team_dummies = pd.get_dummies(df['team'], prefix='team')
    df = pd.concat(objs=[df, team_dummies], axis=1)
    feature_cols += list(team_dummies.columns)

df['podium'] = (df['position_order'] <= 3).astype(int)
feature_cols.append('podium')

df_clean = df[feature_cols].dropna()

print("\nИтог после очистки и создания признаков:")
print(df_clean.head())
print(f"\nВсего записей после очистки: {len(df_clean)}")

df_clean.to_csv('data/cleaned_f1_data.csv', index=False)

print("\nДанные обработаны и сохранены в 'data/cleaned_f1_data.csv'.")

```

Рисунок 1 – Код для предобработки данных

```

Run  1_preprocessing.x

↑
↓
Исходные данные (первые строки):
   Year Round      Grand Prix ... Q1 Time Q2 Time Q3 Time
0  1950     1 British Grand Prix ...    NaN    NaN    NaN
1  1950     1 British Grand Prix ...    NaN    NaN    NaN
2  1950     1 British Grand Prix ...    NaN    NaN    NaN
3  1950     1 British Grand Prix ...    NaN    NaN    NaN
4  1950     1 British Grand Prix ...    NaN    NaN    NaN

[5 rows x 34 columns]

Всего записей: 25840

Столбцы в исходном датасете: ['Year', 'Round', 'Grand Prix', 'Date', 'Sta

После выбора нужных столбцов:
   position_order  grid  laps      team
0                  1    70 Alfa Romeo
1                  2    70 Alfa Romeo
2                  3    70 Alfa Romeo
3                  4    68 Talbot-Lago
4                  5    68 Talbot-Lago

Столбцы после переименования: ['position_order', 'grid', 'laps', 'team']

Итог после очистки и создания признаков:
   grid  laps  team_AFM ...  team_Wolf  team_Zakspeed  podium
0     1    70    False ...    False       False        1
1     2    70    False ...    False       False        1
2     4    70    False ...    False       False        1
3     6    68    False ...    False       False        0
4     9    68    False ...    False       False        0

[5 rows x 198 columns]

Всего записей после очистки: 23288

Данные обработаны и сохранены в 'data/cleaned_f1_data.csv'.

Process finished with exit code 0

```

Рисунок 2 – Результат выполнения предобработки данных

Код для выполнения задания 1 представлен на рисунке 3. Результат его работы – на рисунке 4. Построенная диаграмма – на рисунке 5.

```

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

df = pd.read_csv('data/cleaned_f1_data.csv')

plt.figure(figsize=(8, 6))
sns.countplot(x='podium', data=df)
plt.title('Баланс классов (0 = Не подиум, 1 = Подиум)')
plt.xlabel('Класс')
plt.ylabel('Количество')

balance = df['podium'].value_counts(normalize=True) * 100
print("\nВыходы о балансе классов:")
print(f"Класс 0 (Не подиум): {balance[0]:.3f}%")
print(f"Класс 1 (Подиум): {balance[1]:.3f}%")

plt.show()

```

Рисунок 3 – Код для построения гистограммы

Run 2\_balance\_histogram

C:\Users\Remsely\dev\mirea\python\

Выводы о балансе классов:

Класс 0 (Не подиум): 86.006%

Класс 1 (Подиум): 13.994%

Process finished with exit code 0

Рисунок 4 – Результат выполнения кода для построения гистограммы

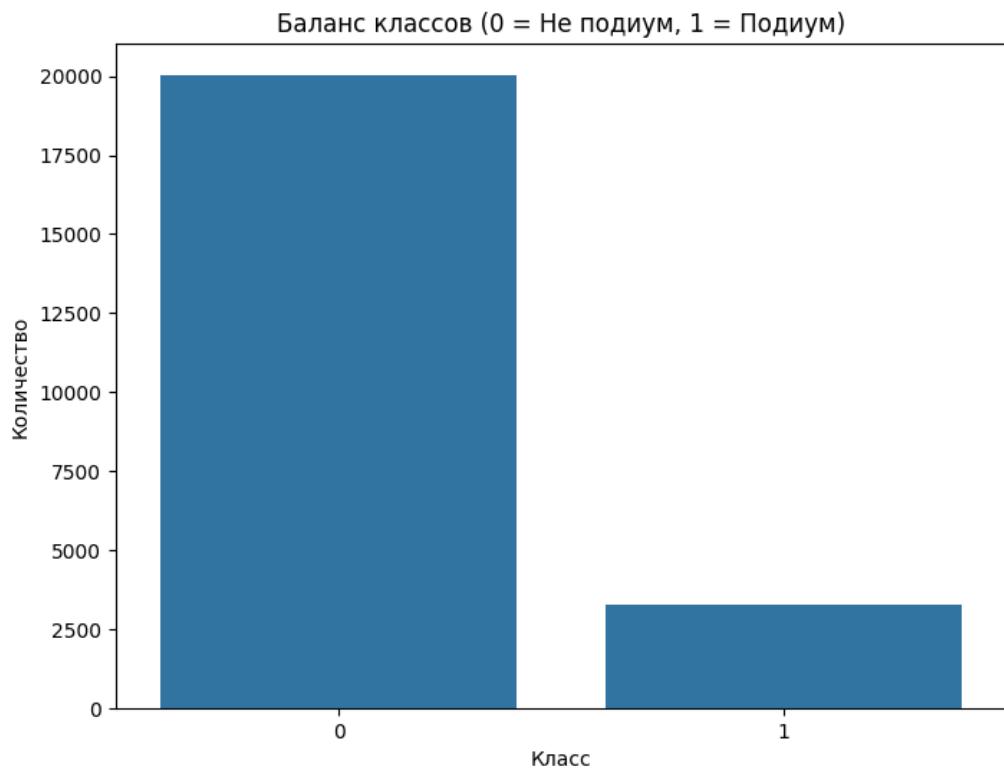


Рисунок 5 – Построенная гистограмма

Вывод: данные сильно несбалансированы. Объекты класса "0" (Не подиум) составляют подавляющее большинство (~86%), в то время как интересующий нас класс "1" (Подиум) является минорным (~14%).

Код для выполнения заданий 3-5 представлен на рисунке 6. Результат его работы – на рисунке 7. Построенные матрицы – на рисунках 8-10.

```

import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

df = pd.read_csv('data/cleaned_f1_data.csv')

X = df.drop(columns=['podium'])
y = df['podium']

X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print(f"Размер X_train: {X_train.shape}")
print(f"Размер X_test: {X_test.shape}")
print(f"Размер y_train: {y_train.shape}")
print(f"Размер y_test: {y_test.shape}")

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

log_reg = LogisticRegression(random_state=42)
svm = SVC(random_state=42)
knn = KNeighborsClassifier(n_neighbors=5)

models = {
    "Logistic Regression": log_reg,
    "Support Vector Machine (SVM)": svm,
    "K-Nearest Neighbors (KNN)": knn
}

for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
    disp.plot()

    plt.title(f'Матрица ошибок: {name}')
    plt.show()

    print(f'\nОтчет о классификации: {name}')
    print(classification_report(y_test, y_pred, target_names=['0 (Не подиум)', '1 (Подиум)']))

print("\nВсе матрицы ошибок показаны.")

```

Рисунок 6 – Код для обучения моделей и сравнения результатов

Run 3-5\_model\_training\_and\_comparison (1) ×

```
C:\Users\Remsely\dev\mirea\python\big-data\.venv\Scripts\python.exe C:/Users/Remsely/dev/mirea/python/big-data/.venv/Scripts/3-5_model_training_and_comparison.py
Размер X_train: (18630, 197)
Размер X_test: (4658, 197)
Размер y_train: (18630,)
Размер y_test: (4658,)

Отчет о классификации: Logistic Regression
precision    recall   f1-score   support
0 (Не подиум)      0.92      0.96      0.94     4006
1 (Подиум)        0.66      0.52      0.58      652

accuracy          0.90
macro avg       0.79      0.74      0.76     4658
weighted avg     0.89      0.90      0.89     4658

Отчет о классификации: Support Vector Machine (SVM)
precision    recall   f1-score   support
0 (Не подиум)      0.92      0.96      0.94     4006
1 (Подиум)        0.67      0.49      0.56      652

accuracy          0.90
macro avg       0.80      0.72      0.75     4658
weighted avg     0.89      0.90      0.89     4658

Отчет о классификации: K-Nearest Neighbors (KNN)
precision    recall   f1-score   support
0 (Не подиум)      0.93      0.95      0.94     4006
1 (Подиум)        0.62      0.54      0.58      652

accuracy          0.89
macro avg       0.77      0.74      0.76     4658
weighted avg     0.88      0.89      0.89     4658

Все матрицы ошибок показаны.

Process finished with exit code 0
```

Рисунок 7 – Отчеты о классификации

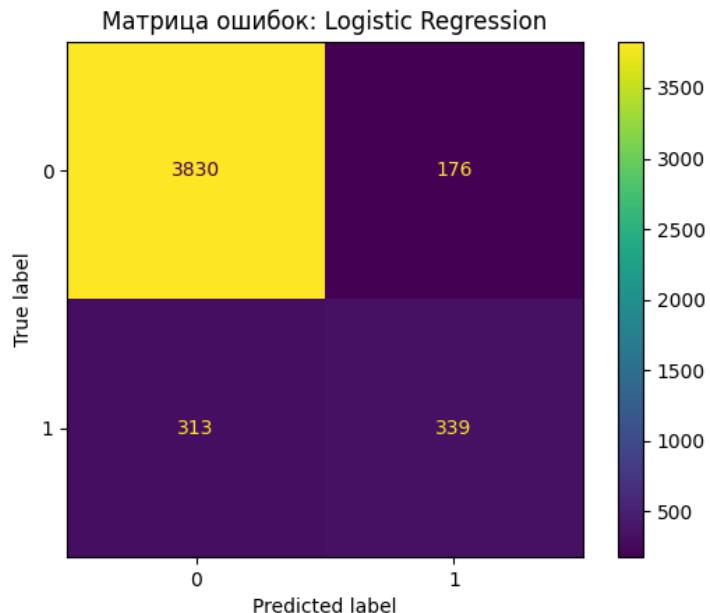


Рисунок 8 – Матрица ошибок для логистической регрессии

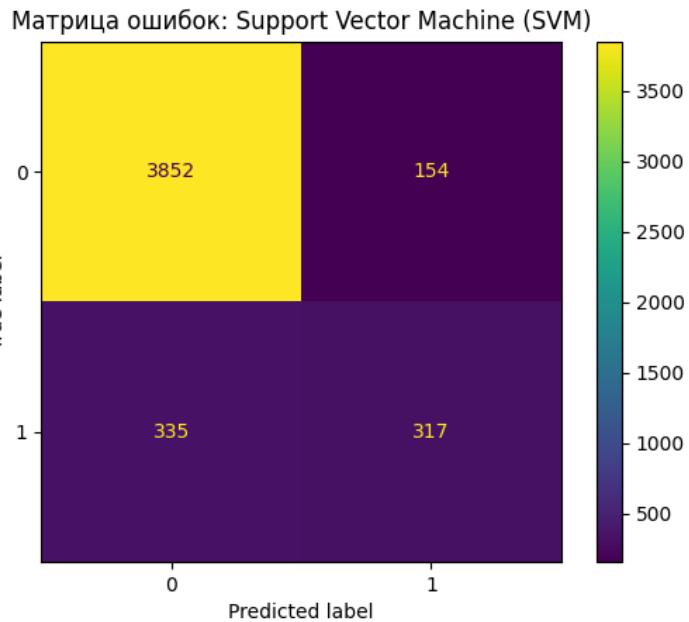


Рисунок 9 – Матрица ошибок для SVM

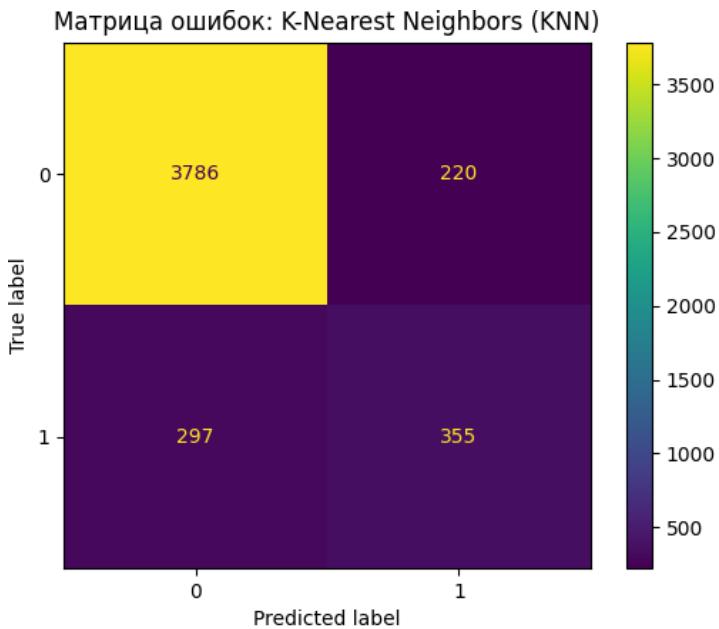


Рисунок 10 – Матрица ошибок для KNN

Выводы: все три модели дают высокую общую точность ( $\sim 0.89\text{--}0.90$ ) за счёт хорошо предсказываемого класса 0 (Не подиум), которого в данных больше всего. Метрики по классу 1 (Подиум) заметно хуже: precision ( $\sim 0.62\text{--}0.67$ ), recall ( $\sim 0.49\text{--}0.54$ ), то есть модели часто ошибаются именно на редких подиумах. Логистическая регрессия и SVM ведут себя очень похоже: немного различаются по числу ложных подиумов и пропущенных подиумов, явного преимущества у SVM нет. KNN даёт ту же общую точность, но чуть лучше находит подиумы (больше истинных 1), при этом чаще ошибочно помечает гонки без подиума как подиум.

## **ВЫВОД**

В ходе выполненной практической работы проведено ознакомление с инструментами классификации данных на Python.