МИНОБРНАУКИ РОССИИ

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

РТУ МИРЭА

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

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №9

по дисциплине «Языки программирования для статистической обработки данных»

Студент группы *ИМБО-11-23, Журавлев Ф.А.*

(подпись)

Преподаватель *Трушин СМ*

(подпись)

Москва 2025 г.

1) ЦЕЛЬ И ЗАДАЧИ

Цель практической работы:

Освоить построение моделей классификации (логистическая регрессия и KNN) с использованием Python, R, а также интерпретацию результатов классификации через отчёты.

Задачи практической работы:

1. Построить модель логистической регрессии:
   - Python: использование библиотеки sklearn и statsmodels.
   - R: функция glm() с семейством binomial.
2. Построить модель K-ближайших соседей (KNN):
   - Python: библиотека sklearn (KneighborsClassifier).
   - R: использование пакета class или caret.
3. Выполнить прогноз на тестовых данных:
   - Python и R: метрики оценки точности (accuracy, precision, recall). 4. Интерпретировать результаты моделей:
   - Выявить важные переменные в логистической регрессии.
   - Оценить точность моделей с помощью ROC-кривой, матрицы ошибок.
   - Сравнить результаты между Python, R

2) РЕЗУЛЬТАТЫ ПРАКТИКИ

Шаг 1) Knn и логистические регрессии.

1.1) Knn и логистические регрессии в Python.

Для более корректной оценки качества классификации необходимо иметь две части данных: Train (для обучения), Test (для проверки качества). Проведем разбиение признаков и целевой переменной. Ниже представлен код:

Рисунок 1.1 — Разделение на train/test.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import seaborn as sbns
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn.model_selection import train_test_split
[9] ✓ 0.1s

df_ = pd.read_csv("../music_data_encoded.csv", sep=',')
df_ = df_.drop('Жанр', axis=1)
X = df_.drop('Код.Жанра', axis=1)
Y = df_['Код.Жанра']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
[10] ✓ 0.0s
```

Далее напишем код, который реализует логистическую регрессию. Сначала с помощью библиотеки sklearn, а замет statsmodels. Ниже представлен код sklear логистической регрессии:

Рисунок 1.2 – логистическая регрессия sklearn.

```
# Обучение модели логистической регрессии
log_reg = LogisticRegression(max_iter=100)
log_reg.fit(x_train, y_train)

# Предсказание на тестовой выборке
y_pred_log_reg = log_reg.predict(x_test)
```

Дальше вычислим метрики логистической регрессии, для изучения ее метрики оценивания: accuracy, recall и precision:

Рисунок 1.3 — Метрики.

```
# Вычисление метрик
accuracy1 = accuracy_score(y_test, y_pred_log_reg)
precision1 = precision_score(y_test, y_pred_log_reg, average='weighted')
recall1 = recall_score(y_test, y_pred_log_reg, average='weighted')

print("accuracy :", accuracy1)
print("precision :", precision1)
print("recall :", recall1)
```

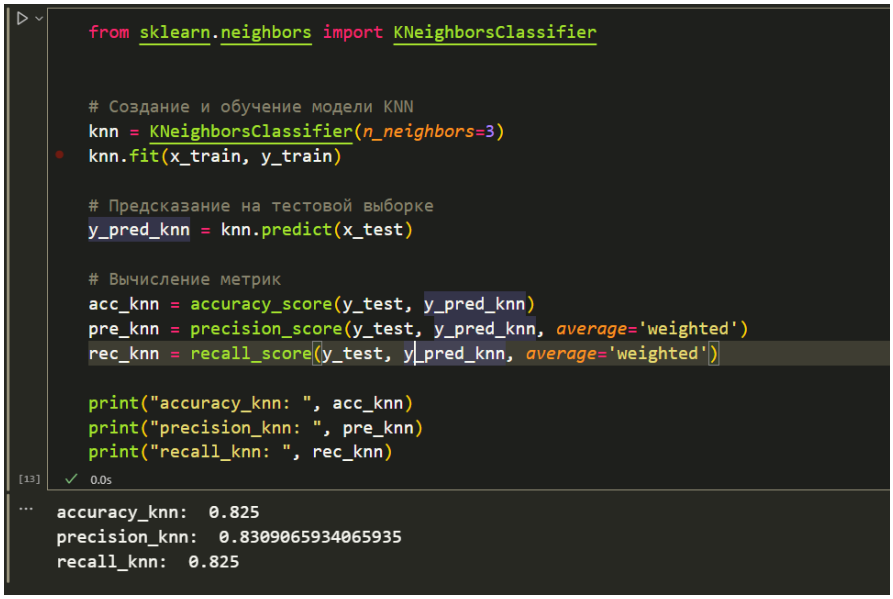
✓ 0.0s

```
accuracy : 0.75
precision : 0.7826388888888889
recall : 0.75
```

Вот какой результат выводит этот код. Можно заметить, что регрессия получилось не очень точной, возможно, из-за того, что для обучения модели требуется больше данных.

Далее построим KNN (метод ближайшего соседа) и вычислим все те же метрики, что делали до этого.

Рисунок 1.6 KNN регрессия и метрики.



```
from sklearn.neighbors import KNeighborsClassifier


# Создание и обучение модели KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)

# Предсказание на тестовой выборке
y_pred_knn = knn.predict(x_test)

# Вычисление метрик
acc_knn = accuracy_score(y_test, y_pred_knn)
pre_knn = precision_score(y_test, y_pred_knn, average='weighted')
rec_knn = recall_score(y_test, y_pred_knn, average='weighted')

print("accuracy_knn: ", acc_knn)
print("precision_knn: ", pre_knn)
print("recall_knn: ", rec_knn)
```

[13] ✓ 0.0s

```
accuracy_knn:  0.825
precision_knn:  0.8309065934065935
recall_knn:  0.825
```

Шаг 2) Knn и логистическая регрессия в R.

2.1) Knn и логистическая регрессия в R.

Рисунок 2.1 — Разделение на train/test.

```
ds <- read.csv(file.choose())
set.seed(42)
trainIndex <- createDataPartition(ds$is_good,p = 0.7,list = FALSE)
ds_train <- ds[trainIndex, ]
ds_test <- ds[-trainIndex, ]
```

Рисунок 2.2 — Логистическая регрессия и метрики

```
model_log1 <- glm(is_good ~ ., data = ds_train,family = 'binomial')
summary(model_log1)

pred_prob <- predict(model_log1,ds_test,type = 'response')
pred_class <- ifelse(pred_prob>=0.5,1,0)
conf_matrix <- table(pred_class,ds_test$is_good)
conf_matrix

accuracy1 <- sum(diag(conf_matrix)) / sum(conf_matrix)
accuracy1
```

Рисунок 2.3 — Показатели регрессии.

model_log1	list [30] (S3: glm, lm)	List of length 30
coefficients	double [5]	-6135.56 46.24 -4.74 5126.91 100.43
residuals	double [70]	1 -1 1 1 -1 -1 ...
fitted.values	double [70]	1.00e+00 2.22e-16 1.00e+00 1.00e+00 2.22e-16 2.22e-16 ...
effects	double [70]	-5.26e-05 2.51e-03 6.46e-04 -1.10e-03 -1.30e-03 1.93e-07 ...
R	double [5 x 5]	-1.52e-04 0.00e+00 0.00e+00 0.00e+00 0.00e+00 -9.09e-04 2.00e-05 0.00e+00 ...
rank	integer [1]	5
qr	list [5] (S3: qr)	List of length 5
family	list [13] (S3: family)	List of length 13
linear.predictors	double [70]	204.9 -232.4 60.5 370.6 -31.2 -289.2 ...
deviance	double [1]	1.696536e-08
aic	double [1]	10
null.deviance	double [1]	96.98345
iter	integer [1]	25
weights	double [70]	2.22e-16 2.22e-16 2.22e-16 2.22e-16 1.28e-13 2.22e-16 ...
prior.weights	double [70]	1 1 1 1 1 1 ...
df.residual	integer [1]	65

Рисунок 2.4 — Матрица и точность.

```
> pred_prob <- predict(model_log1,ds_test,type = 'response')
> pred_class <- ifelse(pred_prob>=0.5,1,0)
> conf_matrix <- table(pred_class,ds_test$is_good)
> conf_matrix

pred_class  0  1
         0 12  1
         1  0 17
>
> accuracy1 <- sum(diag(conf_matrix)) / sum(conf_matrix)
> accuracy1
[1] 0.9666667
```

Далее построим KNN регрессию.

Рисунок 2.5— KNN регрессия.

```
control <- trainControl(method = 'cv',number = 5)
tunegrid <- expand.grid(k = seq(3,15,2))
model_knn <- train(is_good ~ ., data = ds_train,method = 'knn',
                   trControl = control,tunegrid = tunegrid)
model_knn

pred_knn_factor <- factor(pred_knn, levels = c(0, 1))
true_labels_factor <- factor(ds_test$is_good, levels = c(0, 1))

conf_matrix <- confusionMatrix(pred_knn_factor, true_labels_factor)


print(conf_matrix)
```

Рисунок 2.6— Вывод.

```
No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 47, 46, 47
Resampling results across tuning parameters:

  k  RMSE       Rsquared   MAE
  5  0.1064544  0.9534064  0.03743961
  7  0.1433547  0.9131887  0.06116287
  9  0.1589248  0.9005417  0.08333333
```

Рисунок 2.7— Матрица конфузий.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 11  1
         1  0 16

               Accuracy : 0.9643
                 95% CI : (0.8165, 0.9991)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 1.635e-05

                  Kappa : 0.9263

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9412
         Pos Pred Value : 0.9167
```

3

СРАВНЕНИЕ РЕЗУЛЬТАТОВ

Python без всяких проблем поддерживает многоклассовую классификацию, в то время как в R нужно скачивать дополнительные библиотеки, что значительно затрудняет работу.

4 ВЫВОДЫ

В 9 практической работе проведена работа с различными методами классификации: логистическая регрессия и метод ближайших соседей, реализованы в разных языках программирования для статистической обработки данных.