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 №10

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:

Изучить основы машинного обучения, построить простые модели в Python, R, а также освоить методы валидации и анализа ошибок.

Задачи практической работы:

1 . Построить простые модели машинного обучения:

- Python: использование sklearn для создания моделей классификации и регрессии.
- R: применение пакета caret для построения моделей.

2. Провести валидацию моделей:

- Разделение данных на тренировочную и тестовую выборки.
- Оценка качества моделей на тестовых данных.
- Python: метрики accuracy, mean_squared_error.
- R: аналогичные метрики через функции пакета caret.

3. Проанализировать ошибки моделей:

- Построение матрицы ошибок и графиков для анализа ошибок.

4. Сравнить результаты моделей между Python, R.

5. Выявить основные проблемы и сильные стороны инструментов для задач машинного обучения.

2) РЕЗУЛЬТАТЫ ПРАКТИКИ

Шаг 1) Модели машинного обучения.

1.1) Построение моделей машинного обучения в Python

Для начала работы необходимо установить все необходимые библиотеки, так как их достаточно много, то ниже написан код, который показывает все библиотеки, нужные для практической работы.

Рисунок 1.1 — Необходимые библиотеки.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score,recall_score,confusion_matrix,auc, confusion_matrix
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Далее напишем код, с помощью которого мы разделим признаки и целевую для регрессии и классификация, а затем раздели переменные на train и test, для их реализации:

Рисунок 1.2 – Разделение переменных.

```
X_class = ds.drop('is_good',axis = 1)
y_class = ds['is_good']
X_reg = ds.drop('rating',axis = 1)
y_reg = ds['rating']
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X_class,y_class,test_size = 0.3,random_state = 42)
X_train_reg,  X_test_reg,  y_train_reg, y_test_reg =  train_test_split(X_reg,y_reg,test_size = 0.3,random_state= 42)
```

Далее напишем код, с помощью которого будут реализованы логистическая и линейная регресс, а затем посчитаем метрики.

Рисунок 1.3 — Код логистической и линейной регрессии.

```
logreg = LogisticRegression(max_iter= 100)
logreg.fit(X_train_class,y_train_class)
y_pred_class = logreg.predict(X_test_class)
```

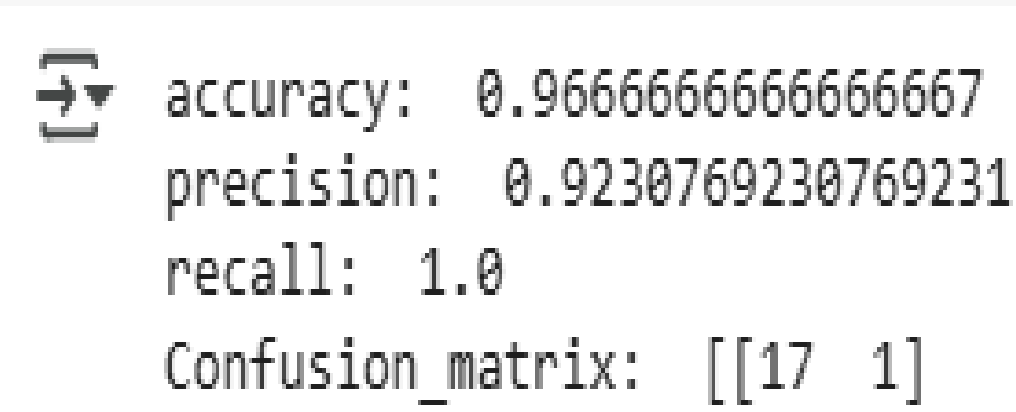
```
[22] linreg = LinearRegression()
     linreg.fit(X_train_reg,y_train_reg)
     y_pred_reg = linreg.predict(X_test_reg)
```

Рисунок 1.4 - Код метрик.

```
acc = accuracy_score(y_test_class,y_pred_class)
prec = precision_score(y_test_class,y_pred_class,average = 'binary')
rec = recall_score(y_test_class,y_pred_class,average='binary')
cm = confusion_matrix(y_test_class,y_pred_class)
print("accuracy: ", acc)
print("precision: ", prec)
print("recall: ", rec)
print("Confusion_matrix: ",cm)
```

Далее посмотрим, какой результат выводит этот код:

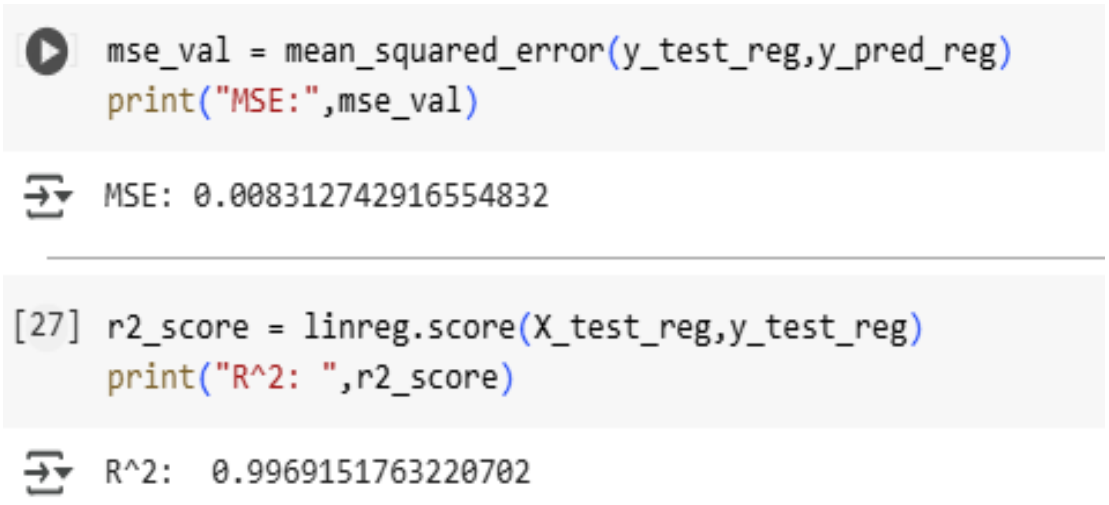
Рисунок 1.5- Метрики.



Проанализируем это позже в 3 параграфе «сравнение результатов».

Далее напишем код, чтобы рассчитать коэффициент детерминации и и среднеквадратичную ошибку. Собственно, R^2 и MSE.

Рисунок 1.6 MSE и R^2.



Шаг 2) Модели машинного обучение в R.

2.1) Построение простейших моделей машинного обучения в Rstuido.

Далее проделаем все то же самое, но уже с помощью языка программирования R, убедившись, что все необходимые пакеты были успешно установлены.

Рисунок 2.1 — Разделение переменных.

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

trainIndex_reg <- createDataPartition(ds$rating,p = 0.7,list = FALSE)
ds_train_reg <- ds[trainIndex_reg, ]
ds_test_reg <- ds[-trainIndex_reg, ]
```

Рисунок 2.2 — Логистическая и линейная регрессия.

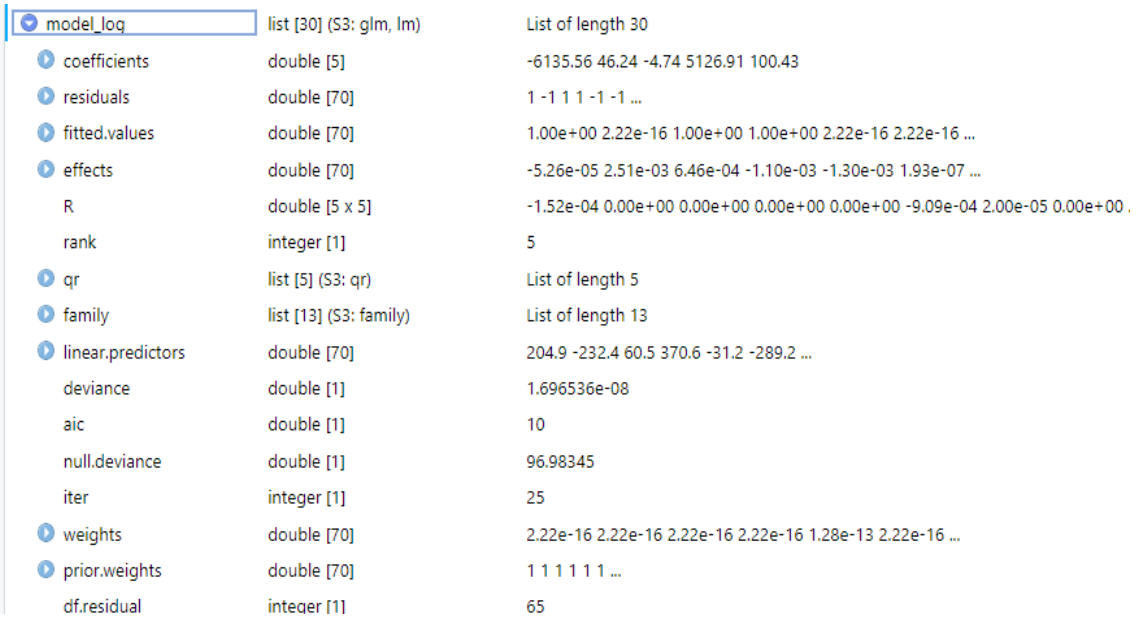
```
model_log <- glm(is_good ~ ., data = ds_train_class,family = 'binomial')
summary(model_log)

model_lin <- lm(rating ~ ., data = ds_test_reg)
summary(model_lin)
pred_reg <- predict(model_lin,newdata = ds_test_reg)
```

На данном этапе мы написали код, для построения обычной логистической модели, а также вычисляет матрицу конфузий и точность.

Рисунок 2.3 —Результаты линейной регрессии.

model_lin	list [12] (S3: lm)	List of length 12
coefficients	double [5]	-15.9426 0.0630 0.0410 19.0306 0.0536
residuals	double [28]	-0.1441 0.1357 -0.1103 -0.0388 0.0886 -0.0251 ...
effects	double [28]	-38.0232 9.0740 -0.9330 -0.3940 0.0761 0.0166 ...
rank	integer [1]	5
fitted.values	double [28]	5.24 6.66 7.21 6.94 8.81 7.73 ...
assign	integer [5]	0 1 2 3 4
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	23
xlevels	list [0]	List of length 0
call	language	lm(formula = rating ~ ., data = ds_test_reg)
terms	formula	rating ~ alcohol + bitterness + original_gravity + is_good
model	list [28 x 5] (S3: data.frame)	A data.frame with 28 rows and 5 columns

Рисунок 2.4 — Результаты логистической регрессии.



model_log	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

Далее рассчитаем все те же коэффициенты:

Рисунок 2.5— Расчет метрик.

```
pred_probs <- predict(model_log, newdata = ds_test_class, type = "response")
pred_class_label <- ifelse(pred_probs > 0.5, 1, 0)  # Преобразование в 0/1

table_pred <- table(pred_class_label,ds_test_class$is_good)
accuracy <- sum(diag(table_pred)) / sum(table_pred)
print(accuracy)

summary(model_lin)$r.squared

mse_val <- mean((ds_test_reg$rating - pred_reg)^2)
print(mse_val)

confusionMatrix(factor(pred_class_label),factor(ds_test_class$is_good))
```

Рисунок 2.6— Вывод метрик.

```
> mse_val <- mean((ds_test_reg$rating - pred_reg)^2)
> print(mse_val)
[1] 0.009460372
> summary(model_lin)$r.squared
[1] 0.9968328
> table_pred <- table(pred_class_label,ds_test_class$is_good)
> accuracy <- sum(diag(table_pred)) / sum(table_pred)
> print(accuracy)
[1] 0.9666667
```

Рисунок 2.7— Матрица конфузий.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 12  1
         1  0 17

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : 4.643e-06

                  Kappa : 0.9315

 Mcnemar's Test P-Value : 1
```

3 СРАВНЕНИЕ РЕЗУЛЬТАТОВ

И там, и там результаты получились одинаковые. Понятное дело работа в R куда удобнее чем в Python, по причине того, что в принципе код куда более компактный, и понятный, единственное, что графики в Python выглядят более красиво, нежели в R.

MSE (среднеквадратичная ошибка): MSE на тестовых данных составляет 0.00946. Это говорит о том, что в среднем предсказания модели отклоняются от фактических значений на небольшую величину. Чем меньше MSE, тем лучше модель соответствует данным.

R-squared (коэффициент детерминации): R-squared равен 0.9968. Это означает, что 99.68% дисперсии целевой переменной объясняется моделью. Это очень высокий показатель, что говорит об отличной подгонке модели к данным. Однако, как и в предыдущем случае, такой высокий R-squared может указывать на переобучение.

Линейная регрессия: Модель линейной регрессии показывает очень хорошие результаты (низкий MSE, высокий R-squared). Однако, необходимо проверить, нет ли переобучения.

Классификация: Модель классификации показывает высокую точность. Но важно оценить и другие метрики (precision, recall, F1-score), особенно если классы несбалансированы.

4 ВЫВОДЫ

В результате 10 практической работы, мы ознакомились со способами реализации линейной регрессии и логистической регрессии. Разобрались в простейших моделях машинного обучения и посчитали самые базовые коэффициенты, перед этим разделив переменные для регрессии и классификации.