Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования «Рыбинский государственный

авиационный технический университет имени П. А. Соловьева»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ УПРАВЛЕНИЯ

Математического и программного обеспечения электронных

вычислительных средств

ОТЧЁТ

по дисциплине:

«Методы и алгоритмы анализа данных»

на тему:

«Логистическая регрессия»

Выполнил: студент группы ИВМ-24 Морозов А. А.

Руководитель: ассистент Вязниковцев Д. А.

Рыбинск 2024

Содержание

[Цель работы 3](#_Toc179403870)

[1. Работа с библиотекой *scikit-learn* 4](#_Toc179403871)

[Вывод **Ошибка! Закладка не определена.**](#_Toc179403872)

# Цель работы

Целями данной лабораторной работы являются:

- изучение теории о логистической регрессии;

- первичная обработка данных;

- бинарная классификация с помощью логистической регрессии.

# 1 Первичная обработка данных

В начале необходимо загрузить датасет о пассажирах Титаника и выбрать данные, которые будем использовать в лабораторной работе:

def download():

df = pd.read\_csv('titanic.csv')

X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]

y = df['Survived']

return X, y

После выполнения кода были созданы 2 *dataframe*.

Далее проанализируем данные на наличие пропусков и заполним недостающими по необходимости с помощью функции *analysisNan*:

def analysisNan(X):

missing = X.isnull().sum()

print(missing[missing > 0].index.tolist())

for i in missing[missing > 0].index.tolist():

if pd.api.types.is\_numeric\_dtype(X[i]):

X[i] = pd.Series(X[i].fillna(X[i].mean(skipna=True)))

else:

X[i] = pd.Series(X[i].fillna("None"))

return X

После этого заменим значение признака *Sex* на числовые 1 и 0 с помощью метода *map* и *lambda*-функции, а также выполним one-hot кодирование:

X['Sex'] = X['Sex'].map(lambda x: 1 if x == 'male' else 0)

X = pd.get\_dummies(X)

Для разделения данных на обучающие и тестовые выборки воспользуемся функцией *srez*():

def srez(X, y):

X\_train, X\_test, Y\_train, Y\_test = X.iloc[:int(len(X) \* 0.7)], X.iloc[int(len(X) \* 0.7):], y.iloc[:int(len(y) \* 0.7)], y.iloc[int(len(y) \* 0.7):]

return X\_train, X\_test, Y\_train, Y\_test

После выполнения функции в консоль выведется размер выборок (623, 10) (268, 10) (623,) (268,)

Для масштабирования данных воспользуемся функцией *mashtab*():

def mashtab(X\_train, Y\_train, X\_test):

print("mashtab")

scaler = StandardScaler()

print(X\_train)

scaler.fit(X=X\_train, y=Y\_train)

scaler\_train = scaler.transform(X=X\_train)

scaler\_test = scaler.transform(X=X\_test)

# 2 Бинарная классификация

Создадим объект класса *LogisticRegression* и обучим его с помощью функции *regression*():

def regression(X\_train, Y\_train):

logreg = LogisticRegression()

logreg.fit(X\_train, Y\_train)

print(f"Коэффициент w{logreg.coef\_},\

\nКоэффициент w0 {logreg.intercept\_}")

return logreg

Получим значения *w*0 и *w* 3.57929185 и [-9.91879184e-01 -2.53548383e+00 -3.21493877e-02 -2.58266244e-01 3.63072952e-02 -1.88868653e-03 1.22646434e+00 7.95724363e-02 1.11431993e+00 7.33393352e-01] (массив) соответственно

Далее отсортируем весовые коэффициенты, создадим новый *dataframe* из полученных значений и создадим график, который покажет самые важные признаки для обученной модели (рисунок 1).

def sortirovka(logreg):

sorted\_weights = sorted(zip(logreg.coef\_.ravel(), X.columns), reverse=True)

print(sorted\_weights)

df = pd.DataFrame(sorted\_weights)

df = df.rename(columns={0: "weights", 1: "features"})

print(df)

plt.barh(df["features"], df["weights"])

plt.show()

return df

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рисунок 1 – График весов признаков

Из рисунка 1 следует то, что самым важным признаком для модели является признак *Sex*

Далее вычисляем предсказание вероятности принадлежности объекта к положительному классу для тестовой части с помощью матричного произведения *dot* и функции *ravel* в *numpy*

def logistic\_function(x):

try:

return (1 / (1 + np.exp(-x)))

except Exception as e:

print(e)

pred\_prob = np.ravel(logistic\_function(np.asarray(np.dot(X\_test, logreg.coef\_.T) + logreg.intercept\_, dtype=float)))

Вычислить предсказание вероятности принадлежности объекта к положительному классу можно и с помощью метода *predict\_proba* и проверить совпадают ли предсказанные значения:

pred\_predict\_proba = logreg.predict\_proba(X\_test)[:, 1]

print(f"pred\_prob равна pred\_predict\_proba? {np.all([pred\_prob, pred\_predict\_proba])}")

В консоли отобразится: «pred\_prob равна pred\_predict\_proba? True»

Чтобы получить в качестве предсказания метки классов, можно предсказанные вероятности принадлежности объекта к положительному классу бинаризовать по порогу (например, по порогу 0.5):

pred\_bin = (pred\_prob >= 0.5).astype(int)

Теперь можно вычислить долю верных ответов у *pred\_bin* и *pred\_class*

print(f'Точность predict\_proba: {np.mean(pred\_bin == Y\_test[:].values)}')

pred\_class = logreg.predict(X\_test)

print(f'Точность predict: {np.mean(pred\_class == Y\_test[:].values)}')

В консоль выведется:

Точность predict\_proba: 0.8097014925373134

Точность predict: 0.8097014925373134

# Вывод

Таким образом, обучение логистической регрессии – настройка параметров *w* и *w*0.

Применение – подсчёт вероятностей принадлежности положительному классу как применение логистической функции к скалярному произведению признаков и параметров.

Посмотрим на распределение целевого (рисунок 2) и предсказываемого значений (рисунок 3).

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 2 – Распределение целевого значения

Изображение выглядит как текст, диаграмма, снимок экрана, График

Автоматически созданное описание

Рисунок 3 – Распределение предсказываемого значения

Также логистическая регрессия возвращает вероятности, так, например если вероятность принадлежности к классу 1 от 0,75 до 0,85, то точность такого предсказания будет крайне высока (на данных для лабораторной работы она равна 0,86)