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

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
```

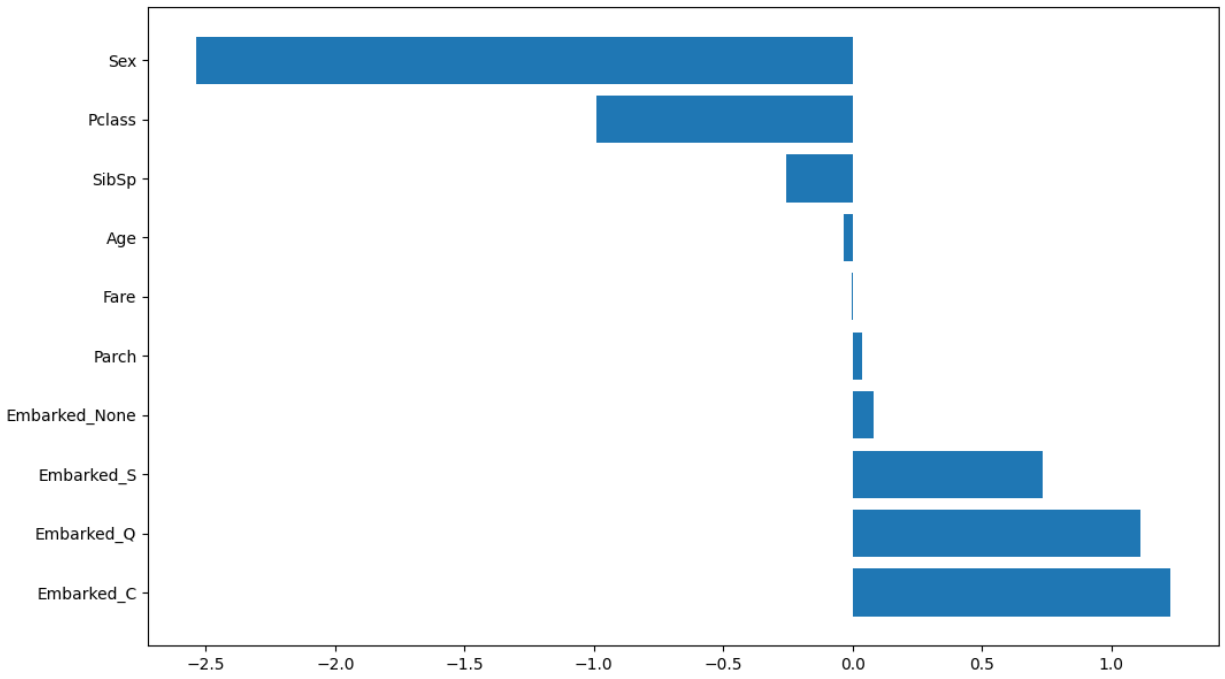

Рисунок 1 – График весов признаков

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).

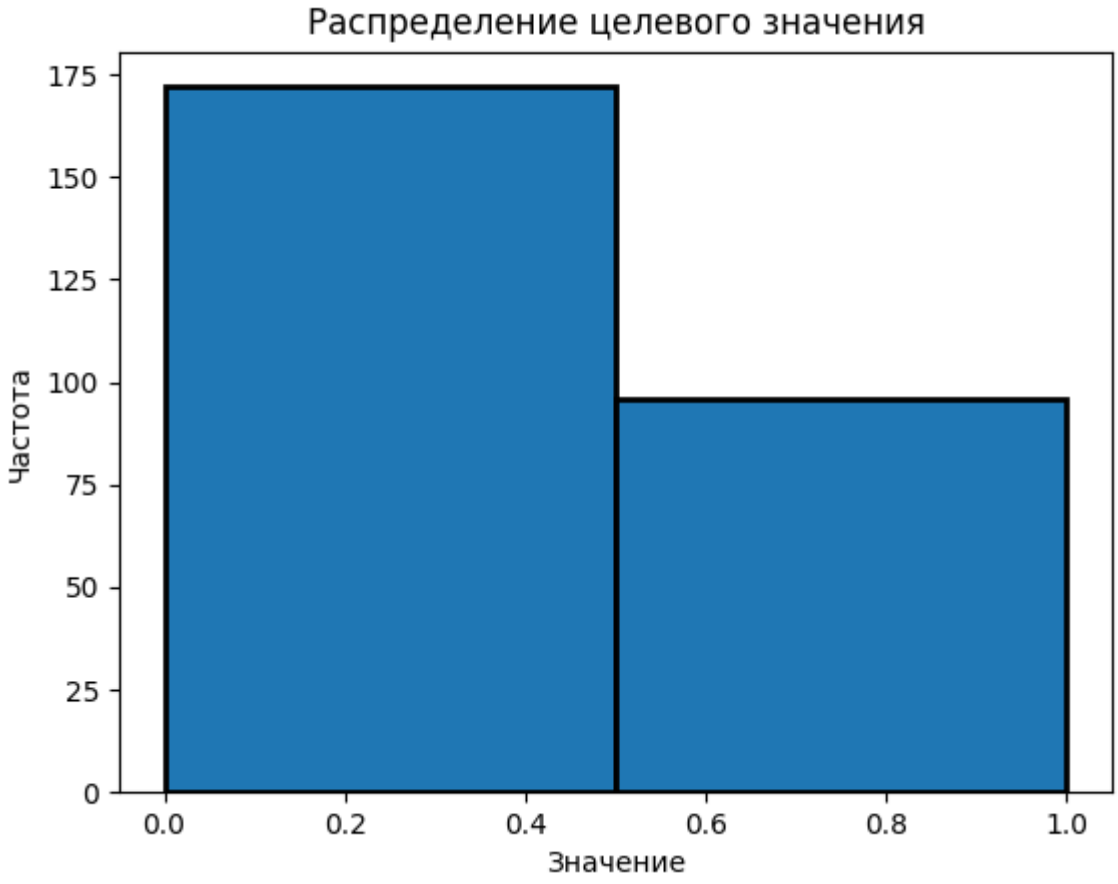

Рисунок 2 – Распределение целевого значения

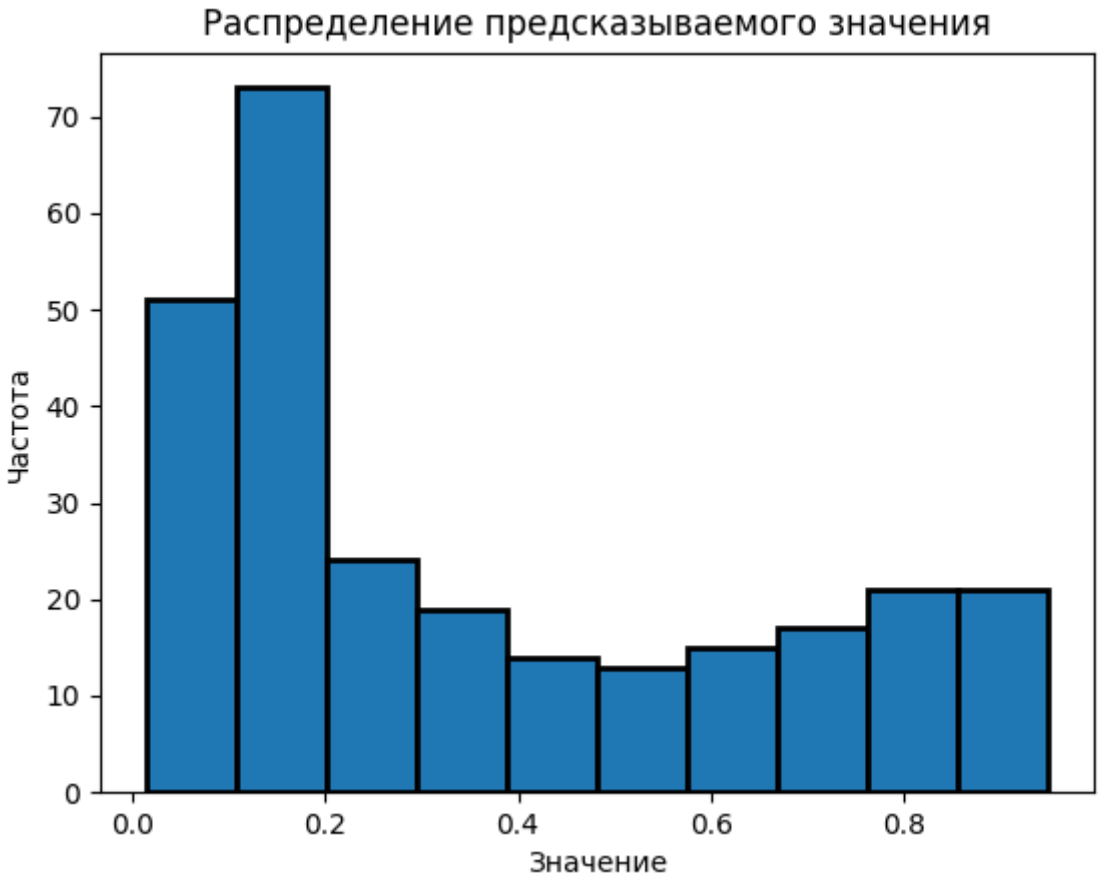

Рисунок 3 – Распределение предсказываемого значения

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)