Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Лабораторная работа №3

по дисциплине

«Методы машинного обучения»

на тему

Выполнила:

студент Ли Лююй

группы ИУ5И-21М

Москва — 2024 г.

1. Цель лабораторной работы

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

2. Задание

1 Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.

2 Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- 2.1 масштабирование признаков (не менее чем тремя способами);
- 2.2 обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
- 2.3 обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
- 2.4 отбор признаков:
- 2.5 один метод из группы методов фильтрации (filter methods);
- 2.6 один метод из группы методов обертывания (wrapper methods);
- 2.7 один метод из группы методов вложений (embedded methods).

3. Ход выполнения работы

```
import pandas as pd

from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

from sklearn.feature_selection import SelectKBest, f_classif

from sklearn.ensemble import RandomForestRegressor

from sklearn.feature_selection import RFE

from sklearn.feature_selection import SelectFromModel


# Загрузка данных

df = pd.read_csv('processed_dataset.csv')
```

```
# Масштабирование признаков

# Метод 1: Стандартизация

scaler_standard = StandardScaler()

df_standard_scaled = pd.DataFrame(scaler_standard.fit_transform(df), columns=df.columns)


# Метод 2: Масштабирование Min-Max

scaler_minmax = MinMaxScaler()

df_minmax_scaled = pd.DataFrame(scaler_minmax.fit_transform(df), columns=df.columns)


# Метод 3: Масштабирование, устойчивое к выбросам

scaler_robust = RobustScaler()

df_robust_scaled = pd.DataFrame(scaler_robust.fit_transform(df), columns=df.columns)


# Обработка выбросов в числовых признаках

# Удаление выбросов: Использование метода IQR

Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

df_no_outliers = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]


# Замена выбросов: Использование случайного леса для регрессии

def replace_outliers_with_predictions(df, column):

    model = RandomForestRegressor()

    train_data = df[df[column].notnull()]

    test_data = df[df[column].isnull()]
```

```
    if not test_data.empty:
        model.fit(train_data.drop(column, axis=1), train_data[column])
        predicted_values = model.predict(test_data.drop(column, axis=1))
        df.loc[df[column].isnull(), column] = predicted_values
    return df

columns_with_outliers = ['Age', 'Income']  # Здесь предполагается, что в этих столбцах есть выбросы
for column in columns_with_outliers:
    df = replace_outliers_with_predictions(df, column)

# Обработка нестандартных признаков
# Предположим, что 'Occupation' является нечисловым некатегориальным признаком, мы преобразуем его в категориальный
if 'Occupation' in df.columns:
    df['Occupation'] = df['Occupation'].astype('category')
    df = pd.get_dummies(df, columns=['Occupation'])

# Выбор признаков
# Фильтрационные методы: Использование SelectKBest и f_classif
selector_kbest = SelectKBest(score_func=f_classif, k=5)
X = df.drop('Income', axis=1)
y = df['Income']
X_kbest = selector_kbest.fit_transform(X, y)

# Обертка: Использование рекурсивного исключения признаков (RFE)
```

```
estimator = RandomForestRegressor()

selector_rfe = RFE(estimator, n_features_to_select=5, step=1)

X_rfe = selector_rfe.fit_transform(X, y)


# Встроенные методы: Использование выбора признаков на основе регуляризации

selector_embedded = SelectFromModel(RandomForestRegressor())

X_embedded = selector_embedded.fit_transform(X, y)


# Экспорт обработанных данных в CSV файл

processed_data_path = 'processed_data.csv'

df.to_csv(processed_data_path, index=False)


print(f"Предварительная обработка данных завершена, файл '{processed_data_path}' сохранен.")
```

Результат: