Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Курсовая работа по дисциплине «Методы машинного обучения» на тему: «Определение диабета на основе диагностических измерений»

ИСПОЛНИТЕЛЬ:

Ханмурзин Т.И. ИУ5-31М

_____

"__"_____2021 г.

Москва 2021

Оглавление

Задание

1. Поиск и выбор набора данных для построения модели машинного обучения. На основе выбранного набора данных строится модель для задачи классификации.
2. Для выбранного датасета решить следующие задачи:
    a. устранение пропусков в данных;
    b. кодирование категориальных признаков;
    c. нормализацию числовых признаков;
    d. масштабирование признаков;
    e. обработку выбросов для числовых признаков;
    f. обработку нестандартных признаков (которые не является числовым или категориальным);
    g. отбор признаков, наиболее подходящих для построения модели;
3. Обучить модель и оценить метрики качества для двух выборок:
    a. исходная выборка, которая содержит только минимальную предобработку данных, необходимую для построения модели (например, кодирование категориальных признаков).
    b. улучшенная выборка, полученная в результате полной предобработки данных в пункте 2.
4. Построить модель с использованием произвольной библиотеки AutoML.
5. Сравнить метрики для трех полученных моделей.

Подготовка данных

Загрузка датасета

Используется датасет пациенток от 21 лет из Национального института диабета, болезней органов пищеварения и почек. Цель набора данных – определить, есть ли у пациента диабет, на основе определенных диагностических измерений, включенных в набор данных.

```
data = pd.read_csv("diabetes.csv", sep=",")
```

```
data.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Устранение пропусков данных

```
data.isnull().sum()
```

```
Pregnancies                 0
Glucose                    14
BloodPressure               0
SkinThickness               0
Insulin                    23
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

С помощью функции impute_column убираем пропуск по каждому признаку.

```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data
```

Колонки заполнены.

```
data.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Обработка категориальных признаков

Датасет не содержит категориальных признаков.

```
data.dtypes
```

```
Pregnancies                 float64
Glucose                     float64
BloodPressure                 int64
SkinThickness                 int64
Insulin                     float64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                         float64
Outcome                       int64
dtype: object
```

Нормализация числовых признаков

```
from sklearn.preprocessing import Normalizer
```

```
normalizer = Normalizer()
n_data = pd.DataFrame(normalizer.fit_transform(data), columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul
n_data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.004408	0.009288	0.899189	0.437106	0.000000	0.006254	0.007830	0.006036	1
1	0.000816	0.005925	0.915476	0.402255	0.000000	0.005499	0.004869	0.002312	0
2	0.007351	0.014364	0.999674	0.000000	0.000000	0.005424	0.010497	0.002864	1
3	0.000842	0.006399	0.944263	0.329061	0.001590	0.005991	0.002389	0.000000	0
4	0.000000	0.012936	0.751619	0.657667	0.003731	0.012070	0.042993	0.003758	1

Масштабирование признаков

```
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()
data[['Pregnancies']] = minmax.fit_transform(data[['Pregnancies']])
data[['Glucose']] = minmax.fit_transform(data[['Glucose']])
data[['Insulin']] = minmax.fit_transform(data[['Insulin']])
data[['BMI']] = minmax.fit_transform(data[['BMI']])
data[['Age']] = minmax.fit_transform(data[['Age']])
```

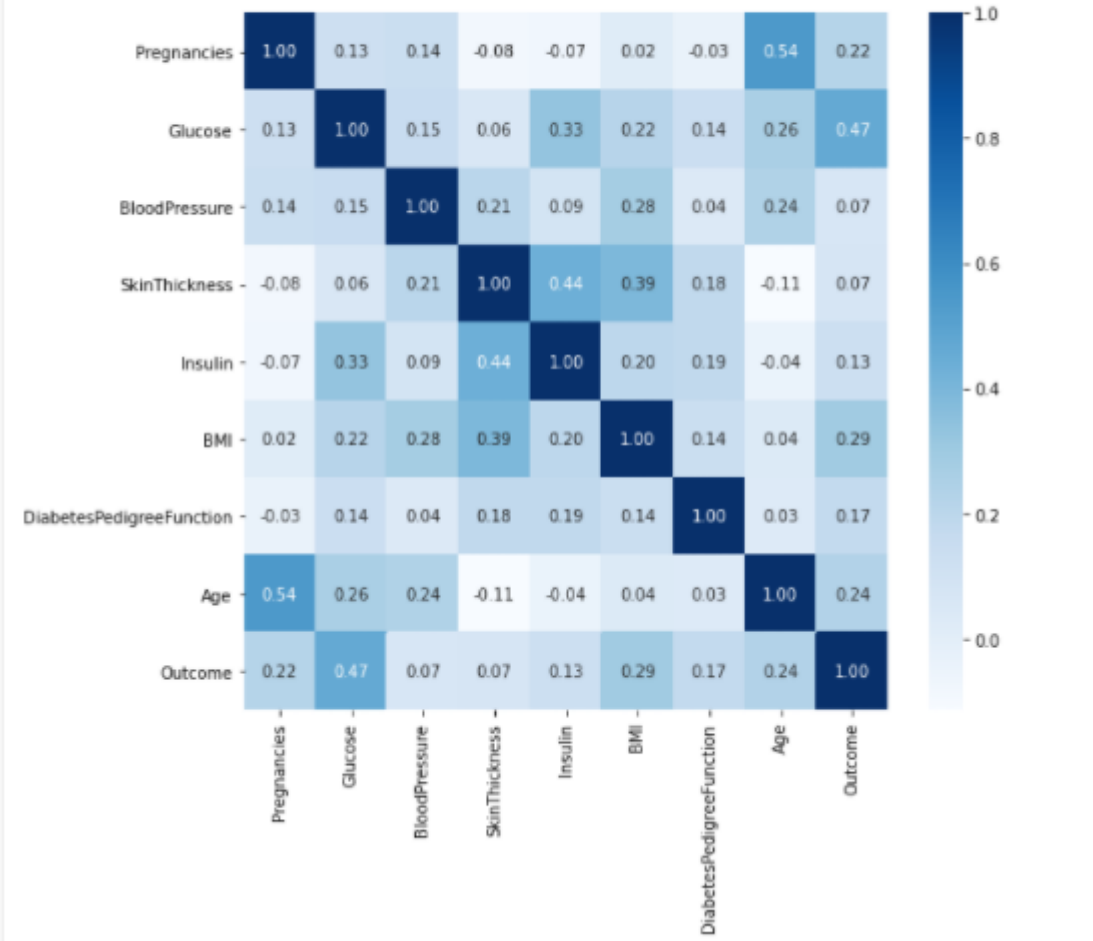
```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	0.226180	0.607510	69.105469	20.536458	0.094326	0.476790	0.471876	0.204015	0.348958
std	0.198210	0.160666	19.355807	15.952218	0.136222	0.117499	0.331329	0.196004	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	0.000000	0.000000
25%	0.058824	0.497487	62.000000	0.000000	0.000000	0.406855	0.243750	0.050000	0.000000
50%	0.176471	0.587940	72.000000	23.000000	0.036052	0.476900	0.372500	0.133333	0.000000
75%	0.352941	0.704774	80.000000	32.000000	0.150414	0.545455	0.626250	0.333333	1.000000
max	1.000000	1.000000	122.000000	99.000000	1.000000	1.000000	2.420000	1.000000	1.000000

Отбор признаков

```
plt.figure(figsize=(9,8))
sns.heatmap(data.corr(), cmap="Blues", annot=True, fmt='.2f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe363afd00>
```



Выберем признаки не отталкиваясь от субъективного мнения с помощью ExhaustiveFeatureSelector.

```
from sklearn.neighbors import KNeighborsRegressor
from mlxtend.feature_selection import ExhaustiveFeatureSelector
```

```
data_x = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
data_y = data['Outcome']
```

```
knn = KNeighborsRegressor(n_neighbors=8)
```

```
efsl = ExhaustiveFeatureSelector(knn, min_features=2, max_features=7, scoring='neg_mean_squared_error', print_progress=True, cv=5
efsl = efsl.fit(data_x, data_y, custom_feature_names=data_x.columns)
```

```
Features: 246/246
```

```
print(efsl.best_feature_names_)
```

```
('Pregnancies', 'Glucose', 'BMI', 'DiabetesPedigreeFunction', 'Age')
```

Результат работы моделей

При подготовленных данных:

LinR – 87,5232%

KNeighborsClassifier – 79,4363%

Tree – 88,2384%

GB – 87,6372%

RandomForestClassifier – 88,2785%

Если данные не подготовлены:

LinR – 72,4783%

KNeighborsClassifier – 63,8842%

Tree – 73,5374%

GB – 72,9456%

RandomForestClassifier – 76,2345%

AutoML

```
train = data.copy()
```

```
from supervised.automl import AutoML
```

```
automl = AutoML()
automl.fit(train[train.columns[:-1]], train['Outcome'])
```

```
AutoML directory: AutoML_1
The task is binary_classification with evaluation metric logloss
AutoML will use algorithms: ['Baseline', 'Linear', 'Decision Tree', 'Random Forest', 'Xgboost', 'Neural Network']
AutoML will ensemble availabe models
AutoML steps: ['simple_algorithms', 'default_algorithms', 'ensemble']
* Step simple_algorithms will try to check up to 3 models
1_Baseline rmse 0.246799 trained in 0.74 seconds
2_DecisionTree rmse 0.212039 trained in 26.57 seconds
3_Linear rmse 0.189151 trained in 3.2 seconds
* Step default_algorithms will try to check up to 3 models
4_Default_Xgboost rmse 0.124482 trained in 3.93 seconds
5_Default_NeuralNetwork rmse 0.187101 trained in 1.65 seconds
6_Default_RandomForest rmse 0.17165 trained in 3.9 seconds
* Step ensemble will try to check up to 1 model
Ensemble rmse 0.111026 trained in 0.79 seconds

AutoML fit time: 50.96 seconds
AutoML best model: Ensemble
```

Для полностью подготовленного датасета точность Ансамблевой модели составляет ~88,8% при обучении самой AutoML в 50.96 секунд, хороший результат для таких данных.

Выводы

Можно сделать вывод, что предобработка данных очень важна для решения задач машинного обучения, это доказывают результаты оценки точности обучения.

Список источников

1. Методы машинного обучения. Лекции. Московский государственный технический университет имени Н.Э.Баумана. Автор: Гапанюк Ю.Е. - 2021