Практическая работа № 3

**Прогнозирование заболеваний**

**1. Постановка задачи**

**Цель**: Построить модель, предсказывающую развитие диабета у пациентов на основе клинических и демографических данных.  
**Датасет**: [Pima Indians Diabetes Dataset](https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)

* **Признаки**:
  + Pregnancies (число беременностей),
  + Glucose (уровень глюкозы),
  + BloodPressure (артериальное давление),
  + SkinThickness (толщина кожи),
  + Insulin (инсулин),
  + BMI (индекс массы тела),
  + DiabetesPedigreeFunction (генетическая предрасположенность),
  + Age (возраст).
* **Целевая переменная**: Outcome (0 — нет диабета, 1 — диабет).

**Проблемы**:

* Дисбаланс классов (≈35% пациентов с диабетом).
* Пропущенные значения (например, нули в Glucose или Insulin).

**2. Подготовка данных**

**Шаги**:

1. **Загрузка данных**:

import pandas as pd

data = pd.read\_csv("diabetes.csv")

1. **Обработка пропусков**:

* Замена нулей в числовых признаках на NaN и заполнение медианами:

cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

data[cols] = data[cols].replace(0, np.nan)

data.fillna(data.median(), inplace=True)

1. **Нормализация данных**:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = scaler.fit\_transform(data.drop('Outcome', axis=1))

y = data['Outcome']

1. **Борьба с дисбалансом (SMOTE)**:

from imblearn.over\_sampling import SMOTE

smote = SMOTE(random\_state=42)

X\_res, y\_res = smote.fit\_resample(X, y)

1. **Разделение на выборки**:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_res, y\_res, test\_size=0.2, stratify=y\_res)

**3. Построение модели**

**Алгоритмы**:

* Логистическая регрессия,
* Случайный лес (Random Forest),
* Градиентный бустинг (XGBoost).

**Пример для XGBoost**:

import xgboost as xgb

from sklearn.metrics import classification\_report, roc\_auc\_score

model = xgb.XGBClassifier(

objective='binary:logistic',

n\_estimators=200,

max\_depth=5,

learning\_rate=0.1,

subsample=0.8,

colsample\_bytree=0.8,

random\_state=42

)

model.fit(X\_train, y\_train)

**4. Обучение и оценка модели**

**Метрики**:

* **Accuracy**, **Precision**, **Recall**, **F1-Score**,
* **ROC-AUC** (важно для дисбаланса).

**Код для оценки**:

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1]

print(classification\_report(y\_test, y\_pred))

print(f"ROC-AUC: {roc\_auc\_score(y\_test, y\_proba):.2f}")

**Пример вывода**:

precision recall f1-score support

0 0.87 0.82 0.84 100

1 0.83 0.88 0.85 100

accuracy 0.85 200

macro avg 0.85 0.85 0.85 200

weighted avg 0.85 0.85 0.85 200

ROC-AUC: 0.93

**Визуализация**:

* Матрица ошибок:

from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred)

* ROC-кривая:

from sklearn.metrics import RocCurveDisplay

RocCurveDisplay.from\_predictions(y\_test, y\_proba)

**5. Интерпретация модели**

**Важность признаков**:

import matplotlib.pyplot as plt

xgb.plot\_importance(model)

plt.show()

**SHAP-значения**:

import shap

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(X\_test)

shap.summary\_plot(shap\_values, X\_test, feature\_names=data.columns[:-1])

**6. Анализ ошибок**

* **Ложноположительные случаи**: Пациенты без диабета, ошибочно классифицированные как больные.
* **Ложноотрицательные случаи**: Пропущенные случаи диабета (критично для медицины).

**Рекомендации**:

1. Увеличить вес класса 1 (диабет) в функции потерь.
2. Использовать ансамбли моделей (Stacking).
3. Добавить клинические данные (например, уровень HbA1c).

**7. Пример улучшения модели**

**Оптимизация гиперпараметров (Optuna)**:

import optuna

def objective(trial):

params = {

'max\_depth': trial.suggest\_int('max\_depth', 3, 10),

'learning\_rate': trial.suggest\_float('learning\_rate', 0.01, 0.3),

'subsample': trial.suggest\_float('subsample', 0.6, 1.0),

'colsample\_bytree': trial.suggest\_float('colsample\_bytree', 0.6, 1.0),

}

model = xgb.XGBClassifier(\*\*params)

model.fit(X\_train, y\_train)

return roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1])

study = optuna.create\_study(direction='maximize')

study.optimize(objective, n\_trials=50)

print(f"Лучшие параметры: {study.best\_params}")

**8. Выводы**

1. Прогнозирование заболеваний требует тщательной обработки данных (пропуски, дисбаланс).
2. XGBoost показывает высокую точность и интерпретируемость.
3. Метрики вроде ROC-AUC и Recall критичны для медицинских задач.
4. Оптимизация гиперпараметров и ансамблирование улучшают качество моделей.