Лабораторная работа №2. Классификация

ГОРБАН АРТЕМИЙ М8О-307Б-23

ДАТАСЕТ: TITANIC-DATASET

Цель работы:

- Обработать датасет Titanic
- Обучить и оценить CatBoost
- Сравнить метрики при разных настройках
- Показать работу с признаками

Подготовка данных:

Исходные данные: 891 пассажир, 11 признаков

Проблемы: пропуски в Age (177), Cabin (687), дисбаланс классов (38% выжило)

Обработка:

- Заполнение Age медианой по группам (Pclass + Sex)
- Извлечение Title из Name (Mr, Miss, Mrs, Master)
- Создание FamilySize, IsAlone, Deck из Cabin
- Удалены PassengerId, Name, Ticket

```
df_raw['Sex'] = df_raw['Sex'].map({'male': 0, 'female': 1})

df_raw['Embarked'] = df_raw['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

df_raw['CabinLetter'] = df_raw['Cabin'].str[0]
df_raw['CabinLetter'] = df_raw['CabinLetter'].fillna('Unknown')
cabin_mapping = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'T': 7, 'Unknown': 8}
df_raw['CabinCode'] = df_raw['CabinLetter'].map(cabin_mapping)
df_raw['HasCabin'] = (~df_raw['Cabin'].isna()).astype(int)

print(f'Типы признаков:\n{df_raw.dtypes}')
```

```
Типы признаков:
Survived         int64
Pclass           int64
Name            object
Sex              int64
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked       float64
CabinLetter     object
CabinCode        int64
HasCabin         int64
dtype: object
```

Feature Engineering

Title - социальный статус (сильная корреляция с выживанием)

FamilySize = SibSp + Parch + 1

IsAlone = (FamilySize == 1)

Deck - первая буква Cabin (A-G, Unknown)

FarePerPerson = Fare / FamilySize

Итог: 20+ признаков вместо исходных 11

```
df_raw['FamilySize'] = df_raw['SibSp'] + df_raw['Parch'] + 1
df_raw['IsAlone'] = (df_raw['FamilySize'] == 1).astype(int)
df_raw['AgeGroup'] = pd.cut(df_raw['Age'], bins=[0, 12, 18, 30, 50, 100], labels=[0, 1, 2, 3, 4])
df_raw['IsChild'] = (df_raw['Age'] < 12).astype(int)
df_raw['FarePerPerson'] = df_raw['Fare'] / df_raw['FamilySize']
df_raw['LogFare'] = np.log1p(df_raw['Fare'])
df_raw['FareGroup'] = pd.qcut(df_raw['Fare'], 4, labels=[0, 1, 2, 3])
df_raw['CabinLetter'] = df_raw['Cabin'].str[0]
df_raw['CabinLetter'] = df_raw['CabinLetter'].fillna('Unknown')

cabin_mapping = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'T': 7, 'Unknown': 8}
df_raw['CabinCode'] = df_raw['CabinLetter'].map(cabin_mapping)
df_raw['Title'] = df_raw['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)

title_mapping = {'Mr': 1, 'Miss': 2, 'Mrs': 3, 'Master': 4, 'Dr': 5, 'Rev': 6, 'Other': 0}
df_raw['Title'] = df_raw['Title'].replace(['Col', 'Major', 'Countess', 'Lady', 'Jonkheer', 'Don', 'Dona', 'Mme', 'Ms', 'Mlle', 'Capt', 'Sir'], 'Other')
df_raw['TitleCode'] = df_raw['Title'].map(title_mapping)
df_raw['ClassSexCode'] = df_raw['Pclass'] * 10 + df_raw['Sex']
df_raw['ClassFareInteraction'] = df_raw['Pclass'] * df_raw['Fare']
df_raw['AgeSexInteraction'] = df_raw['Age'] * df_raw['Sex']

cols_to_drop = ['Name', 'Ticket', 'PassengerId', 'Cabin', 'CabinLetter', 'Title']
df_raw.drop([col for col in cols_to_drop if col in df_raw.columns], axis=1, inplace=True)
df_raw.head(3)
```

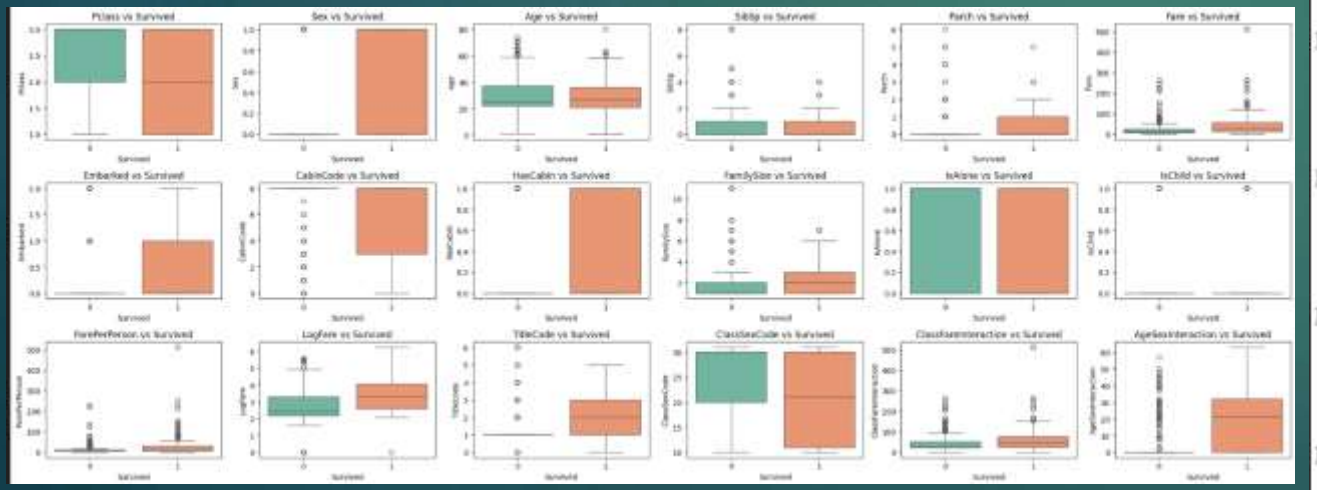
Визуализация данных

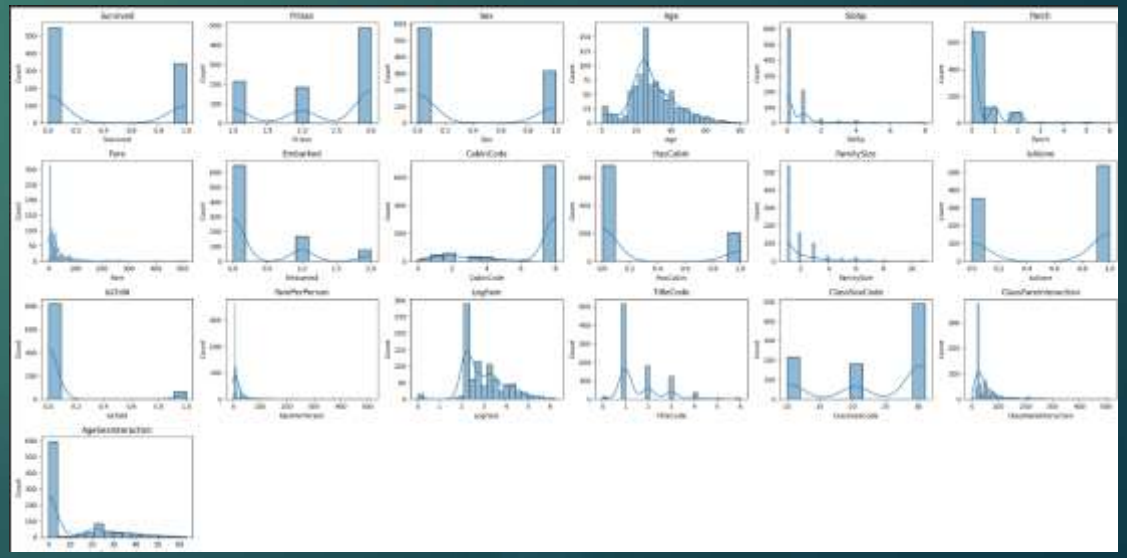
Распределение классов: 62% не выжили, 38% выжили

Возраст vs Выживание: дети <10 лет имели высокий шанс

Класс билета: 1-й класс выживал в 63% случаев, 3-й - только 24%

Пол: 74% женщин выжили vs 19% мужчин





Настройка CatBoost

Базовые параметры:

iterations: 1000

learning_rate: 0.03

depth: 6

loss_function: Logloss

cat_features: ['Sex', 'Embarked', 'Title', 'Deck']

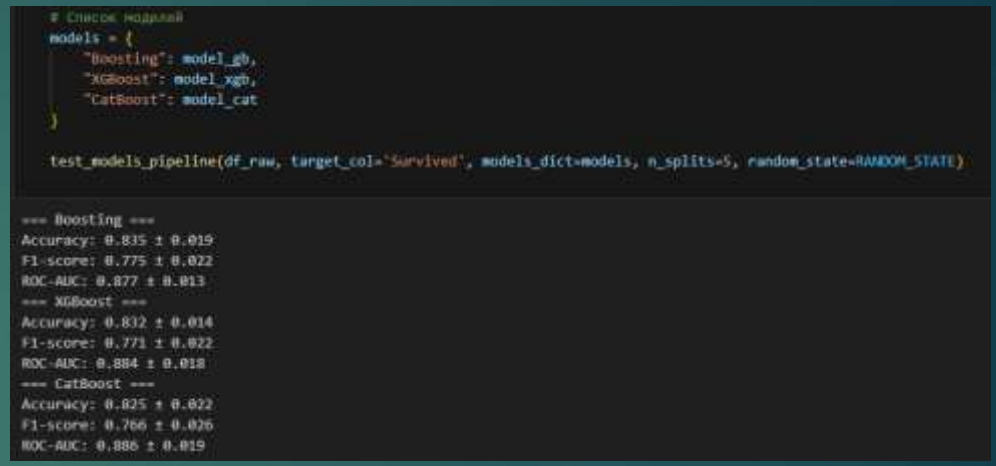
Стратегия валидации:
Stratified 5-Fold Cross-Validation

```python
from catboost import CatBoostClassifier

model_cat = CatBoostClassifier(
    iterations=1000,          # Количество деревьев (итераций бустинга).
                              # Увеличение может повысить точность, но и время обучения.

    learning_rate=0.1,        # Скорость обучения.
                              # Малые значения (0.01–0.1) делают обучение стабильнее, но дольше.

    depth=6,                  # Глубина деревьев.
                              # Контролирует сложность модели (типично 4–10).

    loss_function='Logloss', # Функция потерь (для бинарной классификации).
                              # 'Logloss' — бинарная, 'MultiClass' — многоклассовая.

    l2_leaf_reg=3.0,          # Коэффициент L2-регуляризации.
                              # Повышение снижает переобучение.

    random_seed=RANDOM_STATE, # Для воспроизводимости.

    bootstrap_type='Bayesian', # Метод подвыборки данных при обучении.
                               # 'Bayesian' — по умолчанию, 'Bernoulli', 'MVS' — альтернативы.

    # subsample=0.8,            # Доля данных, используемая для каждого дерева.
                              # Меньше 1.0 — для стохастичности и борьбы с переобучением.

    verbose=0,              # Как часто выводить прогресс обучения (итерации).
                              # 0 — без вывода.

    cat_features=None,        # Список индексов категориальных признаков.
                              # Можно не указывать, если CatBoost сам их определяет.

    task_type='CPU'           # Тип устройства для обучения: 'CPU' или 'GPU'.
)
```

Метрики качества

```python
df_lin['Survived'] = df_lin['Survived'].astype(int)
models = {
    "SVM": model_svm,
    "Logistic Regression": model_logreg,
    "KNN": model_knn
}

test_models_pipeline(df_lin, target_col='Survived', models_dict=models, n_splits=5, random_state=RANDOM_STATE)
```

```
=== SVM ===
Accuracy: 0.824 ± 0.014
F1-score: 0.752 ± 0.024
ROC-AUC: 0.870 ± 0.014
=== Logistic Regression ===
Accuracy: 0.814 ± 0.017
F1-score: 0.750 ± 0.028
ROC-AUC: 0.868 ± 0.020
=== KNN ===
Accuracy: 0.820 ± 0.019
F1-score: 0.760 ± 0.026
ROC-AUC: 0.857 ± 0.017
```



```python
# Список моделей
models = {
    "Boosting": model_gb,
    "XGBoost": model_xgb,
    "CatBoost": model_cat
}

test_models_pipeline(df_raw, target_col='Survived', models_dict=models, n_splits=5, random_state=RANDOM_STATE)
```

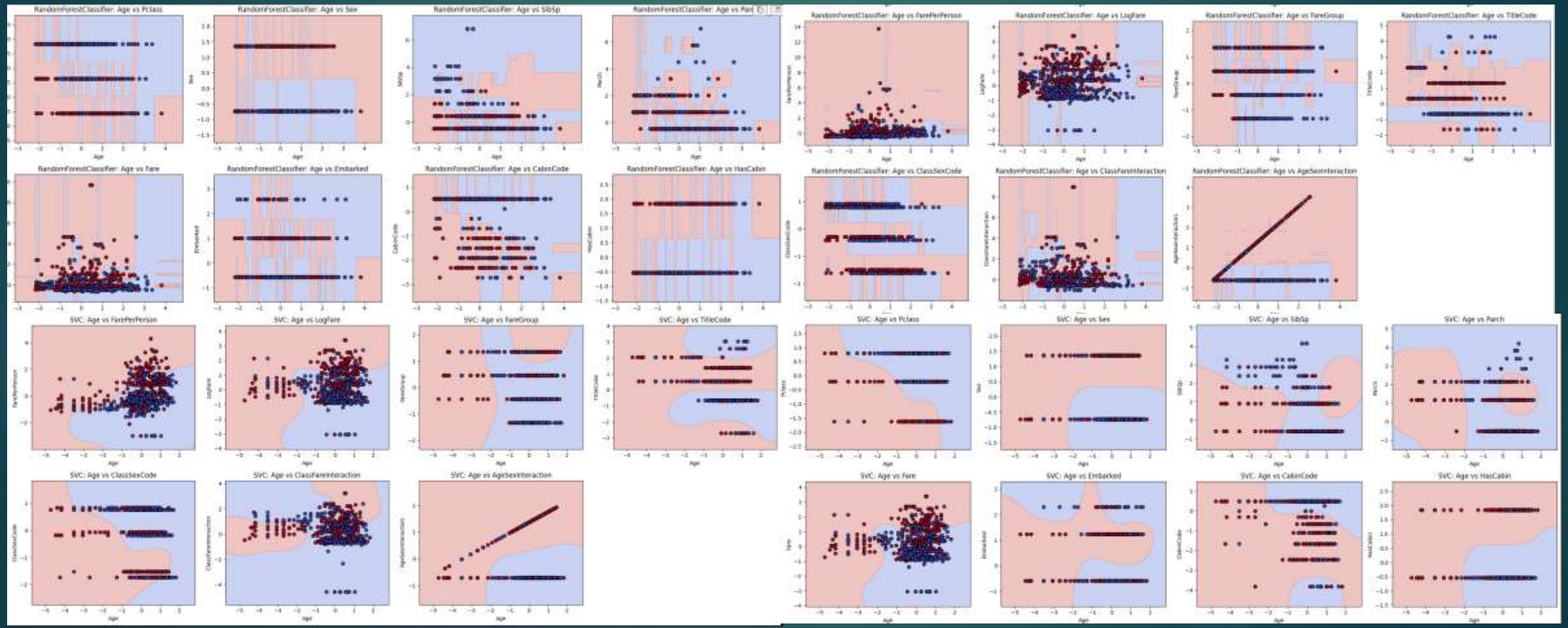
```
=== Boosting ===
Accuracy: 0.835 ± 0.019
F1-score: 0.775 ± 0.022
ROC-AUC: 0.877 ± 0.013
=== XGBoost ===
Accuracy: 0.832 ± 0.014
F1-score: 0.771 ± 0.022
ROC-AUC: 0.884 ± 0.018
=== CatBoost ===
Accuracy: 0.825 ± 0.022
F1-score: 0.766 ± 0.026
ROC-AUC: 0.886 ± 0.019
```

```python
# Список моделей
models = {
    "DesicionTree": model_tree,
    "RandomForest": model_forest
}

test_models_pipeline(df_raw, target_col='Survived', models_dict=models, n_splits=5, random_state=RANDOM_STATE)
```

```
=== DesicionTree ===
Accuracy: 0.768 ± 0.030
F1-score: 0.698 ± 0.042
ROC-AUC: 0.751 ± 0.033
=== RandomForest ===
Accuracy: 0.828 ± 0.024
F1-score: 0.771 ± 0.031
ROC-AUC: 0.880 ± 0.024
```

Поверхности взаимоотношений

Нарисованы графики взаимоотношений признаков:



Важность признаков

Топ-5 самых важных:

Sex (самый значимый)

Pclass (класс билета)

Fare (стоимость)

Title (извлечен из Name)

Age (возраст)

Итоги

Что сделано:

Качественная обработка данных (фичи Title, Deck, FamilySize)

CatBoost показал стабильно высокие результаты

Главный вывод: Качественный feature engineering дал больший прирост, чем тонкая настройка гиперпараметров.