РК2 РТ5-61Б Бабасанова Надежда

Вариант 1

Задание

Для заданного набора данных (load_iris из scikit_learn) постройте модели классификации или регрессии (классификации). Для построения моделей используйте методы 1 и 2 (дерево решений, градиентный бустинг).

Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик).

- Какие метрики качества Вы использовали и почему?

- Какие выводы Вы можете сделать о качестве построенных моделей?

Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Код и скриншоты

Загрузка данных

```
data = load_iris()
data_df = pd.DataFrame(data = data.data, columns =
data.feature_names)
data_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
data_df["target"] = data.target
data_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

```
data_df.isnull().sum()
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

Пропусков нет, все признаки числовые.

Масштабирование

```
data_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(with_mean=False)

scaler = StandardScaler()

scaler.fit(data_df.drop('target', axis=1))
scaled_features = scaler.transform(data_df.drop('target', axis=1))

df_feat = pd.DataFrame(scaled_features, columns=data_df.columns[:-1])
df_feat.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

```
df_feat.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	-1.468455e-15	-1.823726e-15	-1.610564e-15	-9.473903e-16
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.433947e+00	-1.567576e+00	-1.447076e+00
25%	-9.006812e-01	-5.923730e-01	-1.226552e+00	-1.183812e+00
50%	-5.250608e-02	-1.319795e-01	3.364776e-01	1.325097e-01
75%	6.745011e-01	5.586108e-01	7.627583e-01	7.906707e-01
max	2.492019e+00	3.090775e+00	1.785832e+00	1.712096e+00

```
X = df_feat
y = data_df['target']
y.value_counts()
```

```
target
0    50
1    50
2    50
Name: count, dtype: int64
```

Объекты распределены по классам равномерно.

Обучение моделей

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42, stratify=y)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42, stratify=y)
```

Дерево решений

Подбор гиперпараметров

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

param = {
    'max_depth': range(1, 20),
    'min_samples_split': range(2, 10),
    'criterion': ['gini', 'entropy']
}


tree_s = DecisionTreeClassifier(random_state=42)

grid_search = GridSearchCV(tree_s, param)
```

```
grid_search.fit(X, y)


print("Лучшие параметры:", grid_search.best_params_)
print("Лучшая accuracy (на кросс-валидации):",
grid_search.best_score_)

# Оценка на тестовом наборе
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy на тестовом наборе: {test_accuracy:.4f}")
```

```
Лучшие параметры: {'criterion': 'gini', 'max_depth': 3, 'min_samples_split': 2}
Лучшая accuracy (на кросс-валидации): 0.9733333333333334
Accuracy на тестовом наборе: 0.9600
```

```
tree = DecisionTreeClassifier(min_samples_split=2, criterion='gini',
max_depth=3, random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)
```

Градиентный бустинг

Подбор гиперпараметров

```
param = {
    'max_depth': range(1, 6),
    'min_samples_split': range(2, 6),
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 0.2],
    'criterion': ['friedman_mse', 'squared_error']
}

gb_model = GradientBoostingClassifier(random_state=42)

grid_search = GridSearchCV(
    estimator=gb_model,
    param_grid=param,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
)

grid_search.fit(X_train, y_train)

print("Лучшие параметры:", grid_search.best_params_)
print("Лучшая accuracy (на кросс-валидации):",
grid_search.best_score_)

# Оценка на тестовом наборе
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Accuracy на тестовом наборе: {test_accuracy:.4f}")
```

```
Лучшие параметры: {'criterion': 'friedman_mse', 'learning_rate': 0.01, 'max_depth': 1, 'min_samples
_split': 2, 'n_estimators': 50}
Лучшая accuracy (на кросс-валидации): 0.95
Accuracy на тестовом наборе: 0.9400
```

```
grad = GradientBoostingClassifier(
    criterion='friedman_mse',
    n_estimators=50,
    learning_rate=0.01,
    max_depth=1,
    min_samples_split=2,
    random_state=0
).fit(X_train, y_train)

y_pred_gb = grad.predict(X_test)
```

Оценка моделей

```
print("Accuracy дерева решений:", accuracy_score(y_test, y_pred_tree))
print("Accuracy град.бустинга:", accuracy_score(y_test, y_pred_gb))
```

```
Accuracy дерева решений: 0.98
Accuracy град.бустинга: 0.94
```

```
print("f1-мера дерева решений:", f1_score(y_test, y_pred_tree,
average=None))
print("f1-мера град.бустинга:", f1_score(y_test, y_pred_gb,
average=None))
```

```
f1-мера дерева решений: [1.         0.96969697 0.97142857]
f1-мера град.бустинга: [1.         0.91428571 0.90909091]
```

Для оценки были выбраны метрики: accuracy, f1-мера (average=None) для показа общей доли угаданных меток и гармонического среднего между precision и recall по каждому классу.

Как видим, модели хорошо справляются с предсказаниями, особенно решающее дерево. По f1-мере видно, что 0-й класс предсказывается на 100% обеими моделями (т.е. он хорошо отделим от остальных семплов); в дереве на втором месте лучше предсказывается 2-й класс, в GB - 1-й класс.