Московский Государственный технический университет имени Н. Э. Баумана

Лабораторная работа № 6 по курсу: «Технология машинного обучения»

Работу выполнил студент группы ИУ5-63

Федорова Антонина_____

Работу проверил:

Гапанюк Ю.Е._____

Москва 2019

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Текст программы с примерами выполнения программы:

В качестве набора данных я выбрала тот же самый датасет, что и был выбран мной в предыдущих лабораторных работах. Следовательно вся предобработка данных будет схожа с предобработкой в предыдущих лабах.

```
[ ]  #Импорт библиотек
     from google.colab import drive
     import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
     from sklearn.metrics import accuracy_score, balanced_accuracy_score
     from sklearn.metrics import precision_score, recall_score
     from sklearn.model_selection import KFold
     from sklearn.ensemble import BaggingClassifier
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import GridSearchCV
     from sklearn.ensemble import RandomForestClassifier
     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline
     sns.set(style="ticks")
```

Модель бэггинга

```
[ ]  bag = BaggingClassifier(DecisionTreeClassifier(random_state=1), n_estimators=10).fit(X_train, y_train)
```

```
[ ]  target_bag = bag.predict(X_test)
```

```
[ ]  def convert_target_to_binary(array:np.ndarray, target:int) -> np.ndarray:
         # Если целевой признак совпадает с указанным, то 1 иначе 0
         res = [1 if x==target else 0 for x in array]
         return res
```

```
[ ]  # Если целевой признак == 6,
     # то будем считать этот случай 1 в бинарном признаке
     bin_y_test = convert_target_to_binary(y_test, 6)
     list(zip(y_test, bin_y_test))[0:15]
```

```
# Конвертация предсказанных признаков
bin_target_bag = convert_target_to_binary(target_bag, 6)
```

```
ac_bag = accuracy_score(y_test, target_bag)
bas_bag = balanced_accuracy_score(bin_y_test, bin_target_bag)
rec_bag = recall_score(bin_y_test, bin_target_bag)
print('''accuracy_score: {0}
balanced_accuracy_score: {1}
recall_score: {2}'''.format(ac_bag, bas_bag, rec_bag))
```

```
accuracy_score: 0.6
balanced_accuracy_score: 0.6507403380704615
recall_score: 0.5398773006134969
```

Случайный лес

Рекомендуется в задачах классификации брать dl=D‾√, а в задачах регрессии dl=D/3 . В библиотеке scikit-learn это признак max_features

```
forest = RandomForestClassifier(random_state=1, max_features = 2).fit(X_train, y_train)
```

```
target_forest = forest.predict(X_test)
```

```
# Конвертация предсказанных признаков
bin_target_forest = convert_target_to_binary(target_forest, 6)
```

```
ac_forest = accuracy_score(y_test, target_forest)
bas_forest = balanced_accuracy_score(bin_y_test, bin_target_forest)
rec_forest = recall_score(bin_y_test, bin_target_forest)
print('''accuracy_score: {0}
balanced_accuracy_score: {1}
recall_score: {2}'''.format(ac_forest, bas_forest, rec_forest))
```

```
accuracy_score: 0.5825
balanced_accuracy_score: 0.6229258885351143
recall_score: 0.5644171779141104
```

Подбор гиперпараметра с использованием GridSearchCV и кросс-валидации

Для бэггинга

```
n_range = np.array(range(1,202,10))
tuned_parameters = [{'n_estimators': n_range}]
tuned_parameters
```

```
[{'n_estimators': array([  1,  11,  21,  31,  41,  51,  61,  71,  81,  91, 101, 111, 121,
        131, 141, 151, 161, 171, 181, 191, 201])}]
```

```
bag_gs = GridSearchCV(BaggingClassifier(DecisionTreeClassifier()), tuned_parameters, cv=KFold(n_splits=3), sco
bag_gs.fit(data[cols_x], data[col_y])
```

```
bag_gs.best_params_
```

```
{'n_estimators': 51}
```

```
bag_gs.best_estimator_.fit(X_train, y_train)
target_bag_gs = bag_gs.best_estimator_.predict(X_test)
```

```
# Конвертация предсказанных признаков
bin_target_bag_gs = convert_target_to_binary(target_bag_gs, 6)
```

```
ac_bag_gs = accuracy_score(y_test, target_bag_gs)
bas_bag_gs = balanced_accuracy_score(bin_y_test, bin_target_bag_gs)
rec_bag_gs = recall_score(bin_y_test, bin_target_bag_gs)
print('''accuracy_score: {0}
balanced_accuracy_score: {1}
recall_score: {2}'''.format(ac_bag_gs, bas_bag_gs, rec_bag_gs))
```

```
accuracy_score: 0.60875
balanced_accuracy_score: 0.6555292381765939
recall score: 0.5705521472392638
```

Для случайного леса

```
[ ] n_range2 = np.array(range(1,5,1))
    tuned_parameters2 = [{'max_features': n_range2}]
    tuned_parameters2
```

```
[{'max_features': array([1, 2, 3, 4])}]
```

```
[ ] forest_gs = GridSearchCV(RandomForestClassifier(), tuned_parameters2, cv=KFold(n_splits=10), scoring='accuracy')
    forest_gs.fit(data[cols_x], data[col_y])
```

```
[ ] forest_gs.best_params_
```

```
{'max_features': 1}
```

CODE TEXT

```
[ ] forest_gs.best_estimator_.fit(X_train, y_train)
    target_forest_gs = bag_gs.best_estimator_.predict(X_test)
```

```
[ ] # Конвертация предсказанных признаков
    bin_target_forest_gs = convert_target_to_binary(target_forest_gs, 6)
```

```
[ ] ac_forest_gs = accuracy_score(y_test, target_forest_gs)
    bas_forest_gs = balanced_accuracy_score(bin_y_test, bin_target_forest_gs)
    rec_forest_gs = recall_score(bin_y_test, bin_target_forest_gs)
    print('''accuracy_score: {0}
    balanced_accuracy_score: {1}
    recall_score: {2}'''.format(ac_forest_gs, bas_forest_gs, rec_forest_gs))
```

```
accuracy_score: 0.60875
balanced_accuracy_score: 0.6555292381765939
recall_score: 0.5705521472392638
```