Рубежный контроль №2

Столярова Ольга РТ5-61Б

Вариант 16

Задание

Задание. Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). 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.

Метод 1 - Дерево решений

Метод 2 - Градиентный бустинг

Импорт библиотек

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
pd.options.display.float_format = '{:.0f}'.format
```

Загрузка данных

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
filename = '/content/drive/MyDrive/RK2.csv'
```

```
df = pd.read_csv(filename, sep=',')
```

```
df.head()
```

	business_id	business_name	business_address	business_city	business_state	busi
0	101192	Cochinita #2	2 Marina Blvd Fort Mason	San Francisco	CA	
1	97975	BREADBELLY	1408 Clement St	San Francisco	CA	
2	92982	Great Gold Restaurant	3161 24th St.	San Francisco	CA	
3	101389	HOMAGE	214 CALIFORNIA ST	San Francisco	CA	
4	85986	Pronto Pizza	798 Eddy St	San Francisco	CA	

5 rows × 23 columns

Обработка пропусков

```
df.isnull().sum()
```

```
business_id                 0
business_name               0
business_address            0
business_city               0
business_state              0
business_postal_code     1018
business_latitude       19556
business_longitude      19556
business_location       19556
business_phone_number   36938
inspection_id               0
inspection_date             0
inspection_score        13610
inspection_type             0
```

```
violation_id                12870
violation_description       12870
risk_category               12870
Neighborhoods (old)         19594
Police Districts            19594
Supervisor Districts        19594
Fire Prevention Districts   19646
Zip Codes                   19576
Analysis Neighborhoods      19594
dtype: int64
```

```
df.shape
```

```
(53973, 23)
```

```
total_count = df.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 53973
```

```
df = df.dropna(axis=0, how='any')
df.shape
```

```
(6566, 23)
```

```
df.head()
```

	business_id	business_name	business_address	business_city	business_state	bu
11	4794	VICTOR'S	210 TOWNSEND St	San Francisco	CA	
172	63652	SFDH - Banquet Main Kitchen	450 Powell St 2nd Floor	San Francisco	CA	
327	328	Miyako	1470 Fillmore St	San Francisco	CA	
372	2684	ERIC'S RESTAURANT	1500 Church St	San Francisco	CA	
397	328	Miyako	1470 Fillmore St	San Francisco	CA	

5 rows × 23 columns

Кодируем категориальные признаки

Удалим колонки, которые не влияют на целевой признак:

```
df = df.drop(columns='business_name')
df = df.drop(columns='business_address')
df = df.drop(columns='business_city')
df = df.drop(columns='business_state')
df = df.drop(columns='business_location')
df = df.drop(columns='business_phone_number')
df = df.drop(columns='violation_description')
```

```
df.shape
```

```
(6566, 16)
```

```
df.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	insp
11	4794	94107	38	-122	4794
172	63652	94102	38	-122	63652
327	328	94115	38	-122	328
372	2684	94131	38	-122	2684
397	328	94115	38	-122	328

Кодируем категориальные признаки

```
df.dtypes
```

```
business_id                int64
business_postal_code      object
business_latitude        float64
business_longitude       float64
inspection_id             object
```

```
inspection_date              object
inspection_score            float64
inspection_type              object
violation_id                 object
risk_category                object
Neighborhoods (old)         float64
Police Districts            float64
Supervisor Districts        float64
Fire Prevention Districts   float64
Zip Codes                   float64
Analysis Neighborhoods      float64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
df_int = le.fit_transform(df['business_postal_code'])
df['business_postal_code'] = df_int
df_int = le.fit_transform(df['inspection_id'])
df['inspection_id'] = df_int
df_int = le.fit_transform(df['inspection_date'])
df['inspection_date'] = df_int
df_int = le.fit_transform(df['inspection_type'])
df['inspection_type'] = df_int
df_int = le.fit_transform(df['violation_id'])
df['violation_id'] = df_int
df_int = le.fit_transform(df['risk_category'])
df['risk_category'] = df_int
df.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspe
11	4794	5	38	-122	
172	63652	1	38	-122	
327	328	13	38	-122	
372	2684	22	38	-122	
397	328	13	38	-122	

Масштабируем числовые данные

```
from sklearn.preprocessing import MinMaxScaler
```

```
sc1 = MinMaxScaler()
df['business_id'] = sc1.fit_transform(df[['business_id']])
```

```
df['business_latitude'] = sc1.fit_transform(df[['business_latitude']])
df['business_longitude'] = sc1.fit_transform(df[['business_longitude']])
df['inspection_score'] = sc1.fit_transform(df[['inspection_score']])
df['Neighborhoods (old)'] = sc1.fit_transform(df[['Neighborhoods (old)']])
df['Police Districts'] = sc1.fit_transform(df[['Police Districts']])
df['Supervisor Districts'] = sc1.fit_transform(df[['Supervisor Districts']])
df['Fire Prevention Districts'] = sc1.fit_transform(df[['Fire Prevention Districts']])
df['Zip Codes'] = sc1.fit_transform(df[['Zip Codes']])
df['Analysis Neighborhoods'] = sc1.fit_transform(df[['Analysis Neighborhoods']])
df.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspe
11	0	5	1	1	
172	1	1	1	1	
327	0	13	1	1	
372	0	22	0	1	
397	0	13	1	1	

Делим выборку на обучающую и тестовую

```
target = df['risk_category']
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(
    df, target, test_size=0.2, random_state=1)
```

```
data_X_train.shape, data_y_train.shape
```

```
((5252, 16), (5252,))
```

```
data_X_test.shape, data_y_test.shape
```

```
((1314, 16), (1314,))
```

```
np.unique(target)
```

```
array([0, 1, 2])
```

Градиентный бустинг

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
```

```
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error


ab1 = AdaBoostClassifier()
ab1.fit(data_X_train, data_y_train)
```

```
AdaBoostClassifier()
```

```
data_y_pred_1 = ab1.predict(data_X_test)


data_y_pred_1_0 = ab1.predict(data_X_train)


accuracy_score(data_y_train, data_y_pred_1_0)
```

```
1.0
```

```
accuracy_score(data_y_test, data_y_pred_1)
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

```
1.0
```

```
f1_score(data_y_train, data_y_pred_1_0, average='weighted')
```

```
1.0
```

```
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"RMSE: {mean_squared_error(y_test, y_pred)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")


print_metrics(data_y_train, data_y_pred_1_0)
```

```
R^2: 1.0
RMSE: 0.0
MAE: 0.0
```

Дерево решений

```
from sklearn.tree import DecisionTreeRegressor


def stat_tree(estimator):
    n_nodes = estimator.tree_.node_count
    children_left = estimator.tree_.children_left
    children_right = estimator.tree_.children_right

    node_depth = np.zeros(shape=n_nodes, dtype=np.int64)
    is_leaves = np.zeros(shape=n_nodes, dtype=bool)
    stack = [(0, -1)]  # seed is the root node id and its parent depth
    while len(stack) > 0:
        node_id, parent_depth = stack.pop()
        node_depth[node_id] = parent_depth + 1

        # If we have a test node
        if (children_left[node_id] != children_right[node_id]):
            stack.append((children_left[node_id], parent_depth + 1))
            stack.append((children_right[node_id], parent_depth + 1))
        else:
            is_leaves[node_id] = True

    print("Всего узлов:", n_nodes)
    print("Листовых узлов:", sum(is_leaves))
    print("Глубина дерева:", max(node_depth))
    print("Минимальная глубина листьев дерева:", min(node_depth[is_leaves]))
    print("Средняя глубина листьев дерева:", node_depth[is_leaves].mean())
```

Построим модель дерева с глубиной = 3

```
regr5 = DecisionTreeRegressor(max_depth=3)
model5 = regr5.fit(data_X_train, data_y_train)
```

```
data_y_pred_2 = model5.predict(data_X_test)
```

Выведем основную статистику для дерева:

```
stat_tree(model5)
```

```
Всего узлов: 5
Листовых узлов: 3
Глубина дерева: 2
Минимальная глубина листьев дерева: 1
Средняя глубина листьев дерева: 1.6666666666666667
```

График важности признаков:

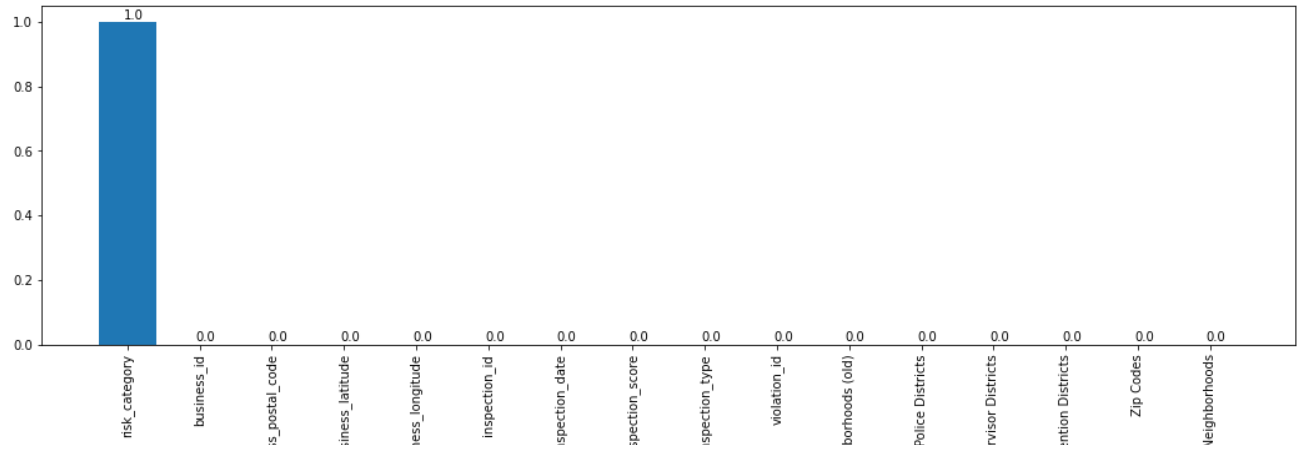
```
from operator import itemgetter
```

```
def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data


list(zip(df.columns.values, model5.feature_importances_))

     [('business_id', 0.0),
      ('business_postal_code', 0.0),
      ('business_latitude', 0.0),
      ('business_longitude', 0.0),
      ('inspection_id', 0.0),
      ('inspection_date', 0.0),
      ('inspection_score', 0.0),
      ('inspection_type', 0.0),
      ('violation_id', 0.0),
      ('risk_category', 1.0),
      ('Neighborhoods (old)', 0.0),
      ('Police Districts', 0.0),
      ('Supervisor Districts', 0.0),
      ('Fire Prevention Districts', 0.0),
      ('Zip Codes', 0.0),
      ('Analysis Neighborhoods', 0.0)]


car_tree_cl_fl_1, car_tree_cl_fd_1 = draw_feature_importances(model5, df)
```




```
accuracy_score(data_y_test, data_y_pred_1)
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_2, average='micro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_2, average='macro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_2, average='weighted')
```

```
1.0
```

```
print_metrics(data_y_test, data_y_pred_2)
```

```
R^2: 1.0
RMSE: 0.0
MAE: 0.0
```

✓ 0 сек. выполнено в 17:41