Контрольная работа №2

Робертс Даниил Александрович

Москва 2022

1 Рубежный контроль №2

1.1 Методы построения моделей машинного обучения.

Задача: 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.

Датасет: https://www.kaggle.com/datasets/fivethirtyeight/fivethirtyeight-comic-characters-dataset?select=dc-wikia-data.csv

[7]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
import xgboost as xgb
import graphviz
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split, GridSearchCV,␣
↪StratifiedKFold
```

[8]:
```
data = pd.read_csv('dc-wikia-data.csv', sep=',')
data.head()
```

[8]:

	page_id	name	urlslug	\
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	
3	1659	James Gordon (New Earth)	\/wiki\/James_Gordon_(New_Earth)	
4	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	

	ID	ALIGN	EYE	HAIR	SEX	\
0	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	

```
1  Secret Identity  Good Characters   Blue Eyes  Black Hair  Male Characters
2  Secret Identity  Good Characters  Brown Eyes  Brown Hair  Male Characters
3  Public Identity  Good Characters  Brown Eyes  White Hair  Male Characters
4  Secret Identity  Good Characters   Blue Eyes  Black Hair  Male Characters

   GSM              ALIVE  APPEARANCES FIRST APPEARANCE    YEAR
0  NaN  Living Characters       3093.0        1939, May  1939.0
1  NaN  Living Characters       2496.0    1986, October  1986.0
2  NaN  Living Characters       1565.0    1959, October  1959.0
3  NaN  Living Characters       1316.0   1987, February  1987.0
4  NaN  Living Characters       1237.0      1940, April  1940.0
```

[9]:
```
# Удаление ненужных колонок
data.drop(columns=['page_id', 'name', 'urlslug'], inplace=True)
```

[10]:
```
data.dtypes
```

[10]:
```
ID                  object
ALIGN               object
EYE                 object
HAIR                object
SEX                 object
GSM                 object
ALIVE               object
APPEARANCES        float64
FIRST APPEARANCE    object
YEAR               float64
dtype: object
```

Посмотрим сколько значений пропущено

[11]:
```
total_rows = data.shape[0]
for col in data.columns:
    null_count = data[data[col].isnull()].shape[0]
    col_type = str(data[col].dtype)
    print(f' {col}, ( {col_type} ) , data loss {null_count / total_rows * 100:.
 ↪2f}%')
```

```
ID, ( object ) , data loss 29.19%
ALIGN, ( object ) , data loss 8.72%
EYE, ( object ) , data loss 52.61%
HAIR, ( object ) , data loss 32.98%
SEX, ( object ) , data loss 1.81%
GSM, ( object ) , data loss 99.07%
ALIVE, ( object ) , data loss 0.04%
APPEARANCES, ( float64 ) , data loss 5.15%
FIRST APPEARANCE, ( object ) , data loss 1.00%
YEAR, ( float64 ) , data loss 1.00%
```

Удалим столбцы EYE и GSM из-за большого количества пропусков

[12]:
```
data.drop(columns=['EYE', 'GSM', 'YEAR'], inplace=True)
```

Удалим строки, содержащие пропуски в столбцах YEAR, FIRST APPEARANCE и ALIGN

[13]:
```
data.dropna(axis=0, how='any', subset=['FIRST APPEARANCE', 'ALIGN',␣
↪'APPEARANCES'], inplace=True)
```

Заполним пропуски в категориальных столбцах при помощих самых часто встречаемых значений

[14]:
```
cat_simp_imp = SimpleImputer(strategy='most_frequent')
cat_data_imputed = cat_simp_imp.fit_transform(data.drop(columns=['APPEARANCES']))
data[data.drop(columns=['APPEARANCES']).columns] = cat_data_imputed
data.isnull().sum()
```

[14]:
```
ID                0
ALIGN             0
HAIR              0
SEX               0
ALIVE             0
APPEARANCES       0
FIRST APPEARANCE  0
dtype: int64
```

Теперь закодируем категориальные признаки

[15]:
```
oe = OrdinalEncoder()
attrs_coded = oe.fit_transform(data.drop(columns=['ALIGN', 'APPEARANCES']))
data[data.drop(columns=['ALIGN', 'APPEARANCES']).columns] = attrs_coded

le = LabelEncoder()
target_coded = le.fit_transform(data[['ALIGN']])
data['ALIGN'] = target_coded
le.classes_
```

```
c:\Users\danib\Documents\pyton\myvenv\lib\site-
packages\sklearn\preprocessing\_label.py:115: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[15]:
```
array(['Bad Characters', 'Good Characters', 'Neutral Characters',
       'Reformed Criminals'], dtype=object)
```

1.2 Разделение выборки на обучающую и тестовую

В качестве уелевого признака возьмем столбец ALIGN

[16]:
```
data_X = data.drop(columns=['ALIGN'])
data_y = data[['ALIGN']]
train_X, test_X, train_y, test_y = train_test_split(data_X, data_y)
```

1.3 Дерево решений

[17]:
```
tree_params_search = {
    'max_depth': [2, 4, 6, 8, 10],
    'min_samples_leaf': [0.04, 0.06, 0.08, 0.1],
    'max_features': [0.2, 0.4, 0.6, 0.8, 1]
}
```

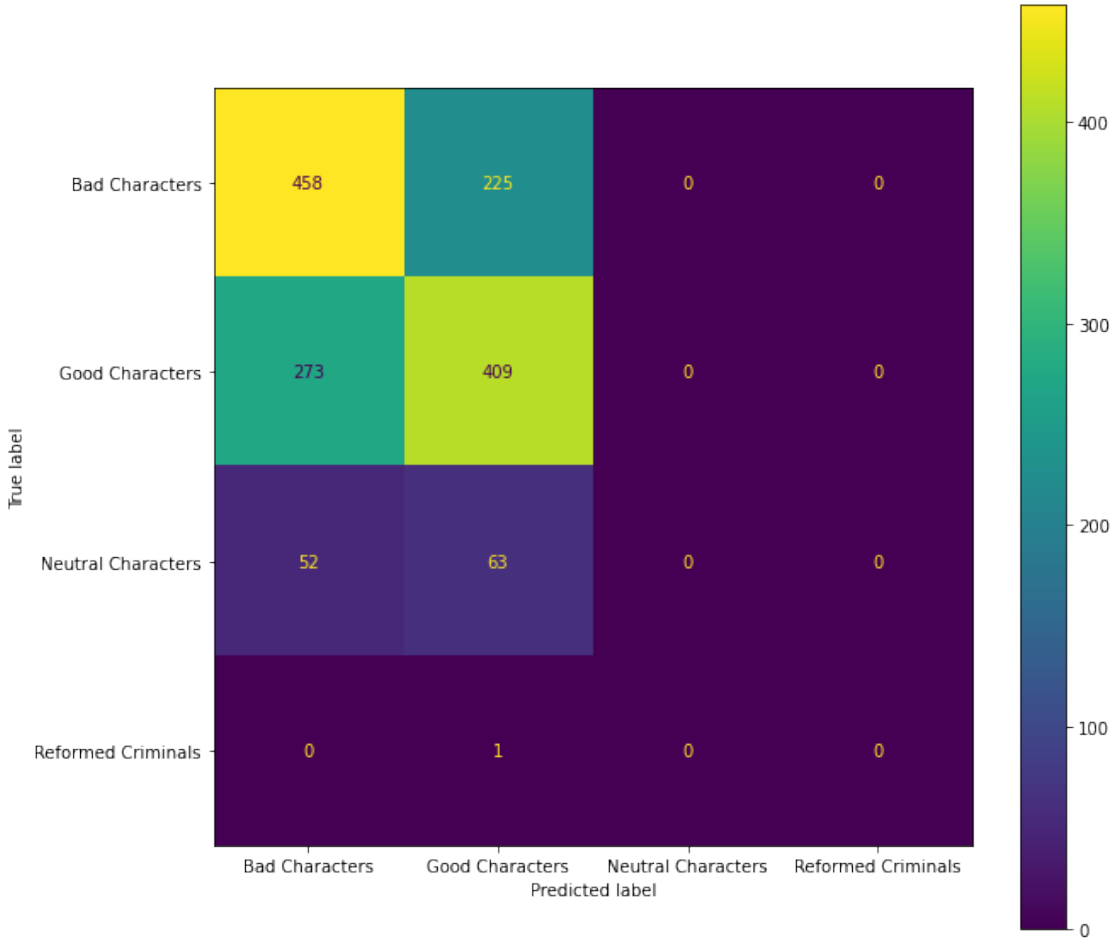
[18]:
```
grid_search_tree = GridSearchCV(DecisionTreeClassifier(), tree_params_search,␣
 ↪scoring='accuracy', cv=StratifiedKFold(n_splits=5), n_jobs=4)
grid_search_tree.fit(train_X, train_y)

grid_search_tree.best_params_, grid_search_tree.best_score_
```

```
c:\Users\danib\Documents\pyton\myvenv\lib\site-
packages\sklearn\model_selection\_split.py:676: UserWarning: The least populated
class in y has only 2 members, which is less than n_splits=5.
  warnings.warn(
```

[18]:
```
({'max_depth': 4, 'max_features': 0.8, 'min_samples_leaf': 0.1},
 0.5905005117603543)
```

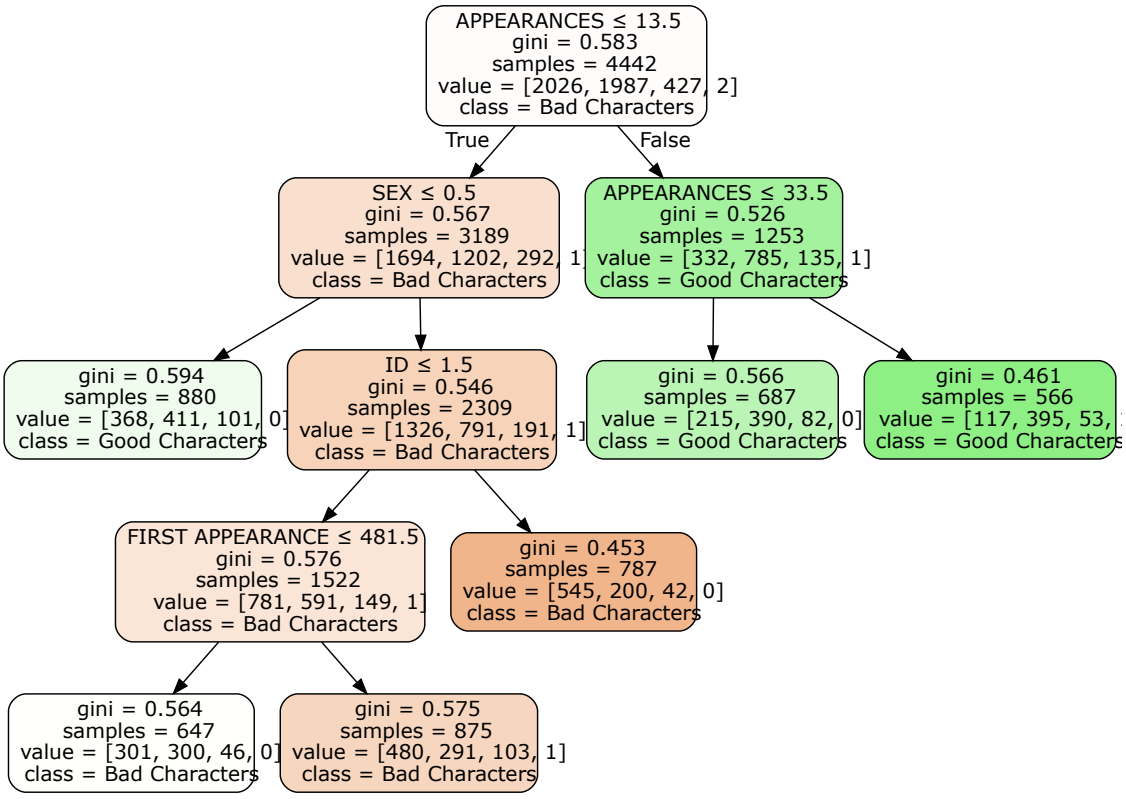
[19]:
```
tree: DecisionTreeClassifier = grid_search_tree.best_estimator_
tree.fit(train_X, train_y)
tree_pred = tree.predict(test_X)
fig, ax = plt.subplots(figsize=(10, 10))
ConfusionMatrixDisplay.from_predictions(test_y, tree_pred, ax=ax,␣
 ↪display_labels=le.classes_)
accuracy_score(tree_pred, test_y)
```

[19]:
```
0.5854152599594868
```



[20]:
```
dot_data = export_graphviz(tree, out_file=None,
                           feature_names=list(data_X.columns),
                           class_names=list(le.classes_),
                           filled=True, rounded=True, special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

[20]:



1.4 Градиентный бустинг

[22]:
```
xgb_model = xgb.XGBClassifier()
xgb_params = {'learning_rate': [0.01, 0.05, 0.1, 0.12, 0.15, 0.2, 0.3, 0.5, 0.7],
              'max_depth': [5, 6, 7, 8],
              'min_child_weight': [2, 3, 5, 11],
              'n_estimators': [1, 2, 5, 10, 12, 15, 20]}

xgb_search = GridSearchCV(estimator=xgb_model, param_grid=xgb_params, cv=5,␣
 ↪n_jobs=4, scoring='accuracy')
xgb_search.fit(train_X, train_y)
xgb_search.best_params_, xgb_search.best_score_
```

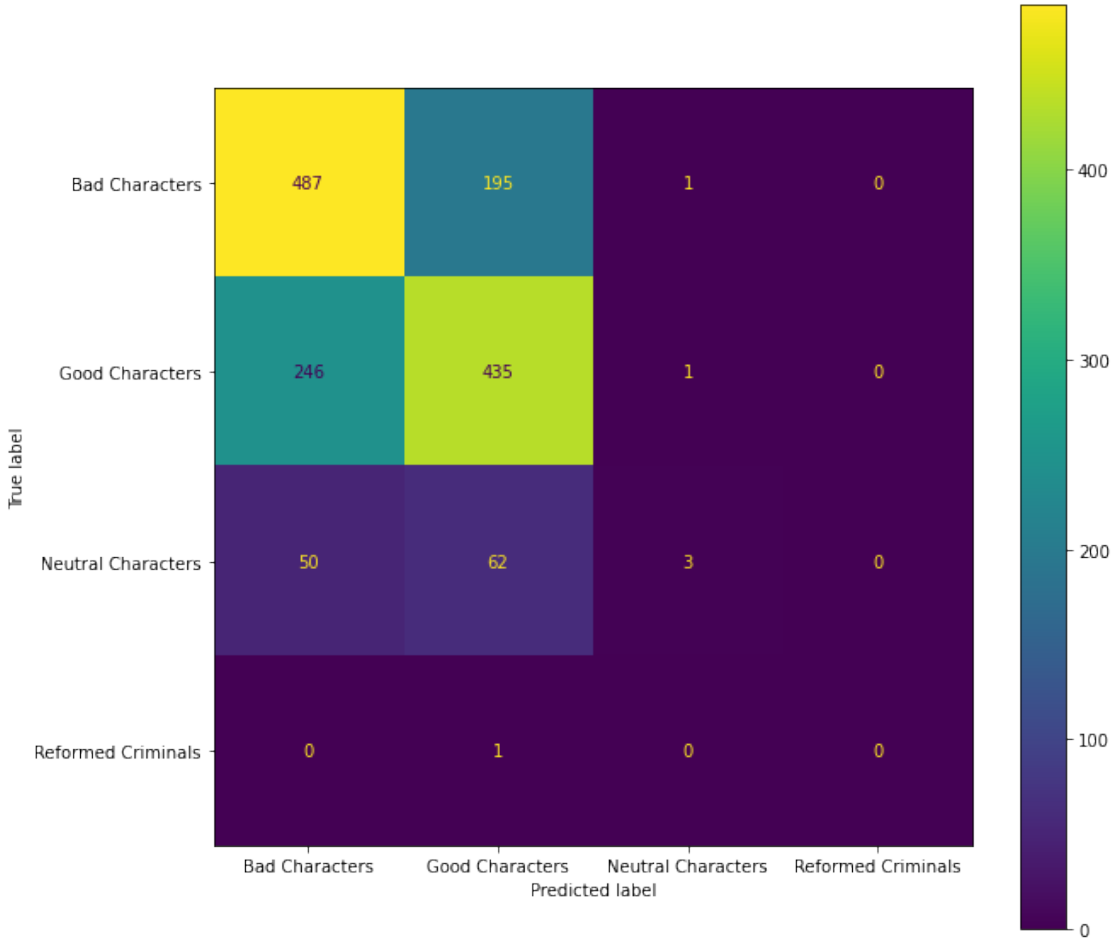
```
c:\Users\danib\Documents\pyton\myvenv\lib\site-
packages\sklearn\model_selection\_split.py:676: UserWarning: The least populated
class in y has only 2 members, which is less than n_splits=5.
  warnings.warn(
```

[22]:
```
({'learning_rate': 0.5,
  'max_depth': 5,
  'min_child_weight': 2,
  'n_estimators': 20},
```

```
0.619091448028456)
```

[23]:
```
xgb_class: xgb.XGBClassifier = xgb_search.best_estimator_
xgb_class.fit(train_X, train_y)
xgb_pred = xgb_class.predict(test_X)
fig, ax = plt.subplots(figsize=(10, 10))
ConfusionMatrixDisplay.from_predictions(test_y, xgb_pred, ax=ax,␣
↪display_labels=le.classes_)
accuracy_score(xgb_pred, test_y)
```

[23]:
```
0.62457798784605
```



Градиентный бустинг получился точнее, чем дерево решений, но при этом все равно ошибается при определении классов Neutral Characters и Reformed Criminals