В этом задании предлагается изучить поведение диких и домашних кошек на основе нескольких характеристик. О кошках имеется некоторая базовая информация (type, group). У кошек были инструкторы. Инструктор обеспечивает кошку питанием, а некоторых кошек обучает проходить полосу препятствий (некоторых - не обучает). Результаты прохождения полосы препятствий оценивались независимо тремя судьями по стобалльной шкале.

Описание столбцов:

- type - тип кошки: дикая (wild) или домашняя (domestic)
- group - закодированная возрастная группа кошки
- education - уровень подготовки инструктора
- meal - тип рациона кошки
- preparation course - обучалась ли кошка прохождению полосы препятствий (проходила ли специальный курс)
- score-1 - балл первого судьи за прохождение кошкой полосы препятствий
- score-2 - балл второго судьи за прохождение кошкой полосы препятствий
- score-3 - балл третьего судьи за прохождение кошкой полосы препятствий

Считайте данные в два pandas dataframe: df_train и df_test.

Задание 1 (0.25 балла). Заполните пропуски в столбце уникальной категорией, если столбец с пропуском категориальный, и средним значением, если столбец числовой. Заполняйте одновременно и df_train, и df_test - одинаковым образом. В ответе укажите количество различных значений, потребовавшихся для заполнения пропусков (это равно количеству новых уникальных категорий плюс количество средних значений для заполнения пропусков в числовых столбцах).

```
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

df_train = pd.read_csv('Data_train.csv')
df_test = pd.read_csv('Data_test.csv')

df_train.head()
df_train.shape
df_train.describe(include='object')
df_train['group'].fillna('Unknown', inplace=True)
```

Задание 2 (0.3 баллов). Кошка прошла полосу препятствий по мнению судьи, если он поставил ей больше 50 баллов. Кошка считается прошедшей полосу препятствий, если все судьи поставили ей больше 50 баллов. В df_train создайте колонку 'Pass' и запишите в неё 1, если кошка прошла полосу препятствий, и 0 иначе. В ответ запишите, сколько кошек из df_train не прошли полосу препятствий.

В df_test от вас скрыта информация о судейских баллах, поэтому неизвестно, прошла кошка полосу препятствия или нет - это и надо будет предсказать в заданиях ниже.

```
df_train['Pass'] = (df_train['score-1'] > 50) & (df_train['score-2'] > 50) & (df_train['score-3'] > 50)
df_train['Pass'] = df_train['Pass'].astype(int)
df_test['Pass'] = 0
# print(df_train.Pass)
len(df_train[df_train['Pass'] == 0])
```

```
145
```

Задание 3 (каждый пункт - 0.25 балла, 1.25 балла максимум).

Это задание выполняйте по данным df_train.

1) Среди всех диких кошек найдите долю кошек, прошедших полосу препятствий. Такую же долю рассчитайте для домашних кошек. В ответе укажите модуль разности этих долей. Ответ округлите до сотых. 2) Сколько кошек среди не прошедших полосу препятствий имели инструктора с уровнем образования "high school"? 3) Сколько диких кошек среди прошедших полосу препятствий не проходили специальный курс подготовки? 4) Чему равна медиана баллов, выставленных первым судьей? 5) Найдите межквартильный размах баллов третьего судьи (третья квартиль минус первая квартиль) для домашних кошек, не проходивших специальный курс подготовки. Комментарий: для вычисления квартилей дискретного распределения используйте интерполяцию меньшим значением (lower interpolation). Это означает, что если искомая квартиль лежит между двумя измерениями i и j, то значение квартили равно i.

```
# Перевод значений 'preparation course' в числовые
df_train['preparation course'] = df_train['preparation course'].apply(lambda x: 1 if x == 'completed' else 0)

# Доля диких и домашних кошек, прошедших полосу препятствий
wild_course_passed = df_train[(df_train['type'] == 'wild') & (df_train['Pass'] == 1)].count() / df_train[df_train['type'] == 'wild'].count()
domestic_course_passed = df_train[df_train['type'] == 'domestic']['Pass'].mean()

# Модуль разности долей
diff = abs(wild_course_passed - domestic_course_passed)
```

```
print(diff)

# Количество кошек среди не прошедших полосу препятствий с
инструктором уровня "high school"
no_course_high_school = df_train[(df_train['Pass'] == 0) &
(df_train['education'] == 'high school')]
print(len(no_course_high_school))

# Количество диких кошек среди прошедших полосу препятствий, не
проходивших специальный курс
wild_passed_no_preparation = df_train[(df_train['type'] == 'wild') &
(df_train['preparation course'] == 0) & (df_train['Pass'] == 1)]
print(len(wild_passed_no_preparation))

# Медиана баллов первого судьи
median_score_1 = df_train['score-1'].median()
print(median_score_1)

# Межквартильный размах баллов третьего судьи для домашних кошек, не
проходивших специальный курс подготовки
domestic_no_course = df_train[(df_train['type'] == 'domestic') &
(df_train['preparation course'] == 0)]
q3 = domestic_no_course['score-3'].quantile(0.75,
interpolation='lower')
q1 = domestic_no_course['score-3'].quantile(0.25,
interpolation='lower')
iqr = q3 - q1
print(iqr)
```

```
type                  0.02258
group                 0.02258
education             0.02258
meal                  0.02258
preparation course    0.02258
score-1               0.02258
score-2               0.02258
score-3               0.02258
Pass                  0.02258
dtype: float64
35
152
66.0
20
```

Задание 4 (0.7 баллов).

а) (0.3 балла). Далее используйте только категориальные столбцы. Закодируйте их с помощью One-hot encoding с учетом того, что мы не хотим получить мультиколлинеарности в новых данных. Сколько получилось числовых столбцов из исходных категориальных? Кодируйте и df_train, и df_test.

б) (0.4 балла). Попытаемся по характеристикам кошки (бывшие категориальные, а теперь - числовые столбцы) предсказать, прошла она полосу препятствий или нет.

Сформируйте из df_train матрицу объект-признак X и вектор ответов y.

Обучите решающее дерево (DecisionTreeClassifier из библиотеки sklearn.tree) глубины 5 с энтропийным критерием информативности на закодированных в пункте а) тренировочных данных по кросс-валидации с тремя фолдами, метрика качества - roc-auc.

Чему равен roc-auc, усредненный по фолдам? Ответ округлите до десятых.

Комментарий: остальные гиперпараметры дерева оставьте дефолтными (splitter='best', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

# Загрузка данных
df_train = pd.read_csv('Data_train.csv')
df_test = pd.read_csv('Data_test.csv')

# Создание столбца 'Pass' в df_train
df_train['Pass'] = ((df_train['score-1'] > 50) &
                    (df_train['score-2'] > 50) &
                    (df_train['score-3'] > 50)).astype(int)

# Сколько кошек не прошли полосу препятствий
not_passed_count = (df_train['Pass'] == 0).sum()
print(not_passed_count)

# а) One-hot encoding для категориальных столбцов
categorical_columns = ['type', 'group', 'education', 'meal',
'preparation course']
encoder = OneHotEncoder(drop='first', sparse_output=False)
```

```
# Применение One-hot encoding к df_train и df_test
encoded_train = encoder.fit_transform(df_train[categorical_columns])
encoded_test = encoder.transform(df_test[categorical_columns])

# Количество новых числовых столбцов
n_new_columns = encoded_train.shape[1]
print(n_new_columns)

# Создание DataFrame из закодированных данных с корректными именами
столбцов
encoded_train_df = pd.DataFrame(encoded_train, index=df_train.index,
columns=encoder.get_feature_names_out(categorical_columns))
encoded_test_df = pd.DataFrame(encoded_test, index=df_test.index,
columns=encoder.get_feature_names_out(categorical_columns))

# Замена категориальных столбцов на закодированные в исходных
DataFrame
df_train_encoded =
pd.concat([df_train.drop(columns=categorical_columns),
encoded_train_df], axis=1)
df_test_encoded =
pd.concat([df_test.drop(columns=categorical_columns),
encoded_test_df], axis=1)

# Преобразование всех имен столбцов в строки
df_train_encoded.columns = df_train_encoded.columns.astype(str)
df_test_encoded.columns = df_test_encoded.columns.astype(str)

# б) Обучение решающего дерева
X = df_train_encoded.drop(columns='Pass')
y = df_train_encoded['Pass']

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5)
scores = cross_val_score(clf, X, y, cv=3, scoring='roc_auc')

# Усредненный roc-auc по фолдам
mean_roc_auc = scores.mean()
print(mean_roc_auc)
```

```
145
13
0.9965986394557823
```

Задание 6 (1.5 балла максимум).

а) (0.25 балла). Подберите глубину решающего дерева (max_depth), перебирая глубину от 2 до 20 с шагом 1 и используя перебор по сетке (GridSearchCV из библиотеки sklearn.model_selection) с тремя фолдами и метрикой качества - roc-auc. В ответ запишите наилучшее среди искомых значение max_depth.

Комментарий: остальные гиперпараметры дерева оставьте дефолтными (splitter='best', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

б) (0.5 балла). Добавьте к данным новый признак cat_bio, содержащий в качестве значений пары значений из столбца type и столбца group. Например, если кошка имеет type='wild' и group='group B', то в cat_bio будет стоять строка '(wild, group B)'. Примените OneHotEncoding (с учетом того, что мы не хотим получить мультиколлинеарности в новых данных) к столбцам 'cat_bio', 'education', 'meal', 'preparation course', а затем обучите решающее дерево глубины 5 с энтропийным критерием информативности на полученных после кодирования данных. Чему равен roc-auc? Ответ округлите до сотых.

Комментарий: остальные гиперпараметры дерева оставьте дефолтными (splitter='best', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.tree import DecisionTreeClassifier

# Загрузка данных
df_train = pd.read_csv('Data_train.csv')
df_test = pd.read_csv('Data_test.csv')

# Создание столбца 'Pass' в df_train
df_train['Pass'] = ((df_train['score-1'] > 50) &
                    (df_train['score-2'] > 50) &
                    (df_train['score-3'] > 50)).astype(int)

# One-hot encoding для категориальных столбцов
categorical_columns = ['type', 'group', 'education', 'meal',
'preparation course']
encoder = OneHotEncoder(drop='first', sparse_output=False)

# Применение One-hot encoding к df_train и df_test
```

```
encoded_train = encoder.fit_transform(df_train[categorical_columns])
encoded_test = encoder.transform(df_test[categorical_columns])

# Создание DataFrame из закодированных данных с корректными именами
столбцов
encoded_train_df = pd.DataFrame(encoded_train, index=df_train.index,
columns=encoder.get_feature_names_out(categorical_columns))
encoded_test_df = pd.DataFrame(encoded_test, index=df_test.index,
columns=encoder.get_feature_names_out(categorical_columns))

# Замена категориальных столбцов на закодированные в исходных
DataFrame
df_train_encoded =
pd.concat([df_train.drop(columns=categorical_columns),
encoded_train_df], axis=1)
df_test_encoded =
pd.concat([df_test.drop(columns=categorical_columns),
encoded_test_df], axis=1)

# Преобразование всех имен столбцов в строки
df_train_encoded.columns = df_train_encoded.columns.astype(str)
df_test_encoded.columns = df_test_encoded.columns.astype(str)

# Подбор max_depth с помощью GridSearchCV
X = df_train_encoded.drop(columns='Pass')
y = df_train_encoded['Pass']

param_grid = {'max_depth': range(2, 21)}
clf = DecisionTreeClassifier(criterion='entropy')
grid_search = GridSearchCV(clf, param_grid, cv=3, scoring='roc_auc')
grid_search.fit(X, y)

best_max_depth = grid_search.best_params_['max_depth']
print(best_max_depth)
```

```
4
```

```
# Добавление нового признака cat_bio
df_train['cat_bio'] = df_train.apply(lambda row: f"({row['type']},
{row['group']})", axis=1)
df_test['cat_bio'] = df_test.apply(lambda row: f"({row['type']},
{row['group']})", axis=1)

# One-hot encoding для новых категориальных столбцов
categorical_columns_new = ['cat_bio', 'type', 'group', 'education',
'meal', 'preparation course']
encoder_new = OneHotEncoder(drop='first', sparse_output=False)

# Применение One-hot encoding к новым категориальным столбцам
encoded_train_new =
```

```
encoder_new.fit_transform(df_train[categorical_columns_new])
encoded_test_new =
encoder_new.transform(df_test[categorical_columns_new])

# Создание DataFrame из закодированных данных с корректными именами
столбцов
encoded_train_df_new = pd.DataFrame(encoded_train_new,
index=df_train.index,
columns=encoder_new.get_feature_names_out(categorical_columns_new))
encoded_test_df_new = pd.DataFrame(encoded_test_new,
index=df_test.index,
columns=encoder_new.get_feature_names_out(categorical_columns_new))

# Замена новых категориальных столбцов на закодированные в исходных
DataFrame
df_train_encoded_new =
pd.concat([df_train.drop(columns=categorical_columns_new),
encoded_train_df_new], axis=1)
df_test_encoded_new =
pd.concat([df_test.drop(columns=categorical_columns_new),
encoded_test_df_new], axis=1)

# Преобразование всех имен столбцов в строки
df_train_encoded_new.columns =
df_train_encoded_new.columns.astype(str)
df_test_encoded_new.columns = df_test_encoded_new.columns.astype(str)

# Обучение решающего дерева на новых данных
X_new = df_train_encoded_new.drop(columns='Pass')
y_new = df_train_encoded_new['Pass']

clf_new = DecisionTreeClassifier(criterion='entropy', max_depth=5)
scores_new = cross_val_score(clf_new, X_new, y_new, cv=3,
scoring='roc_auc')

# Усредненный roc-auc по фолдам
mean_roc_auc_new = scores_new.mean()
print(mean_roc_auc_new)
```

```
0.9965986394557823
```

в) (0.75 балла). Теперь вы можете использовать любую модель машинного обучения для решения задачи. Также можете делать любую другую обработку признаков. Ваша задача - получить наилучшее качество (ROC_AUC).

Качество проверяется на тестовых данных.

ROC_AUC > 0.7 - 0.25 балла

ROC_AUC > 0.74 - 0.75 балла

Сдайте файл result.txt: в файле должна одна колонка с предсказанными значениями целевой переменной для тестовой выборки, без индекса и заголовка.

Во вложении пример файла для отправки результатов.

```
import pandas as pd
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

# Загрузка данных
df_train = pd.read_csv('Data_train.csv')
df_test = pd.read_csv('Data_test.csv')

# Создание столбца 'Pass' в df_train
df_train['Pass'] = ((df_train['score-1'] > 50) &
                    (df_train['score-2'] > 50) &
                    (df_train['score-3'] > 50)).astype(int)

# Добавление нового признака cat_bio
df_train['cat_bio'] = df_train.apply(lambda row: f"({row['type']},
{row['group']})", axis=1)
df_test['cat_bio'] = df_test.apply(lambda row: f"({row['type']},
{row['group']})", axis=1)

# Заполнение NaN значений в категориальных признаках
categorical_columns = ['cat_bio', 'type', 'group', 'education',
'meal', 'preparation course']
for col in categorical_columns:
    df_train[col].fillna('unknown', inplace=True)
    df_test[col].fillna('unknown', inplace=True)

# Подготовка данных для обучения модели
X = df_train.drop(columns=['score-1', 'score-2', 'score-3', 'Pass'])
y = df_train['Pass']

# Разделение данных на тренировочную и валидационную выборки
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Обучение модели CatBoostClassifier
clf = CatBoostClassifier(depth=5, iterations=1000, learning_rate=0.1,
eval_metric='AUC', verbose=200)
clf.fit(X_train, y_train, cat_features=categorical_columns,
eval_set=(X_val, y_val))

# Оценка ROC_AUC на валидационной выборке
y_val_pred = clf.predict_proba(X_val)[:, 1]
roc_auc = roc_auc_score(y_val, y_val_pred)
print(f"ROC_AUC на валидационной выборке: {roc_auc:.2f}")

# Предсказание на тестовых данных
X_test = df_test.drop(columns=['score-1', 'score-2', 'score-3'],
errors='ignore')
y_test_pred = clf.predict_proba(X_test)[:, 1]

# Сохранение предсказаний в файл result.txt
pd.Series(y_test_pred).to_csv('result.txt', index=False, header=False)
```

```
0:    test: 0.6392897 best: 0.6392897 (0)   total: 13.5ms    remaining:
13.5s
200:  test: 0.7358491 best: 0.7577691 (61)  total: 3.18s     remaining:
12.6s
400:  test: 0.7408435 best: 0.7577691 (61)  total: 6.47s     remaining:
9.67s
600:  test: 0.7555494 best: 0.7597114 (560) total: 9.55s     remaining:
6.34s
800:  test: 0.7583241 best: 0.7616537 (697) total: 12.7s     remaining:
3.16s
999:  test: 0.7574917 best: 0.7616537 (697) total: 15.8s     remaining:
0us

bestTest = 0.7616537181
bestIteration = 697

Shrink model to first 698 iterations.
ROC_AUC на валидационной выборке: 0.76
```