Отчет о HW6

imports

In [5]:
```
import os
import json
import pandas as pd
import datetime
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, ShuffleSplit, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, accuracy_score
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier

%matplotlib inline
```

Обработка данных

При обработке данных я удалил записи о матчах с продожительностью менее 10 минут
В части этих игры значения были пусты - это могло быть шумом, т.к. предсказание на этих данных это бросок монетки.

In [2]:
```
# Получаем индексы строк, где 'game_time' больше 600
indices = df_train_features_extended['game_time'] > 600# Фильтруем df_train_feature
df_train_features_after_10min = df_train_features_extended.loc[indices]
y_after_10min = y[indices]
rf_model = RandomForestClassifier(n_estimators=300, max_depth=7, n_jobs=-1, random_
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 2
      1 # Получаем индексы строк, где 'game_time' больше 600
----> 2 indices = df_train_features_extended['game_time'] > 600# Фильтруем df_trai
n_features_extended и y с использованием этих индексов
      3 df_train_features_after_10min = df_train_features_extended.loc[indices]
      4 y_after_10min = y[indices]

NameError: name 'df_train_features_extended' is not defined
```

Я добавил значения по убийствам Roshan в стандартный код "add_new_features"

In [ ]:
```
def add_new_features(df_features, matches_file):
    """
    Аргуенты
    -------
    df_features: таблица с данными
    matches_file: JSON файл с сырыми данными

    Результат
    ---------
    Добавляет новые признаки в таблицу
    """
```

```
    for match in read_matches(matches_file):
        match_id_hash = match['match_id_hash']

        # Посчитаем количество разрушенных вышек обеими командами
        radiant_tower_kills = 0
        dire_tower_kills = 0
        radiant_roshan_kills = 0
        dire_roshan_kills = 0
        for objective in match["objectives"]:
            if objective["type"] == "CHAT_MESSAGE_TOWER_KILL":
                if objective["team"] == 2:
                    radiant_tower_kills += 1
                if objective["team"] == 3:
                    dire_tower_kills += 1
            if objective["type"] == "CHAT_MESSAGE_ROSHAN_KILL":
                if objective["team"] == 2:
                    radiant_roshan_kills += 1
                if objective["team"] == 3:
                    dire_roshan_kills += 1


        df_features.loc[match_id_hash, "radiant_tower_kills"] = radiant_tower_kills
        df_features.loc[match_id_hash, "dire_tower_kills"] = dire_tower_kills
        df_features.loc[match_id_hash, "diff_tower_kills"] = radiant_tower_kills -

        df_features.loc[match_id_hash, "radiant_roshan_kills"] = radiant_roshan_kil
        df_features.loc[match_id_hash, "dire_roshan_kills"] = dire_roshan_kills
        df_features.loc[match_id_hash, "diff_roshan_kills"] = radiant_roshan_kills


        # ... (/¯◡ ‿ ◡)/¯☆*:・゜ добавляем новые признаки ...
```

Randomforest

Используя модифицированные данные я обучил модель

In [3]:
```
# Обучаем модель на данных, исключающих матчи с замерами до 10 минут
rf_model.fit(df_train_features_after_10min.values, y_after_10min)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 # Обучаем модель на данных, исключающих матчи с замерами до 10 минут
----> 2 rf_model.fit(df_train_features_after_10min.values, y_after_10min)

NameError: name 'rf_model' is not defined
```

In [4]:
```
%%time
cv_scores_base = cross_val_score(rf_model, X, y, cv=cv_5, scoring="roc_auc", n_jobs
cv_scores_extended_10 = cross_val_score(rf_model, df_train_features_after_10min.val
                                        cv=cv_5, scoring="roc_auc", n_jobs=-1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
File <timed exec>:1

NameError: name 'rf_model' is not defined
```

print(f"ROC-AUC на кросс-валидации для матчей >600sec:
{cv_scores_extended_10.mean()}")

И получил: ROC-AUC на кросс-валидации для матчей >600sec: 0.8231958297983153

LGBM

Далее я попробовал LGBM

In [ ]:
```
# Создаем экземпляр класса LGBMClassifier
lgb_model = LGBMClassifier(n_estimators=300, max_depth=7, num_leaves=2**7, random_s

# Обучаем модель на данных, исключающих матчи с замерами до 10 минут
lgb_model.fit(df_train_features_after_10min.values, y_after_10min)

# Вычисляем ROC-AUC на кросс-валидации
cv_scores_lgb = cross_val_score(lgb_model, df_train_features_after_10min.values, y_
                                cv=cv_5, scoring="roc_auc", n_jobs=-1)

print(f"ROC-AUC на кросс-валидации для матчей >600sec (LightGBM): {cv_scores_lgb.me
```

И получил значение: ROC-AUC на кросс-валидации для матчей >600sec (LightGBM): 0.7983712955749274

CatBoost

Второй моделью я попробовал catboost

In [ ]:
```
# Создаем экземпляр класса CatBoostClassifier
cb_model = CatBoostClassifier(iterations=300, depth=7, verbose=False, random_state=

# Обучаем модель на данных, исключающих матчи с замерами до 10 минут
cb_model.fit(df_train_features_after_10min.values, y_after_10min)

# Вычисляем ROC-AUC на кросс-валидации
cv_scores_cb = cross_val_score(cb_model, df_train_features_after_10min.values, y_af
                               cv=cv_5, scoring="roc_auc", n_jobs=-1)

print(f"ROC-AUC на кросс-валидации для матчей >600sec (CatBoost): {cv_scores_cb.mea
```

Скор был уже лучше: ROC-AUC на кросс-валидации для матчей >600sec (CatBoost): 0.7977811135791114

После этого я дополнил таблицу данными о общей ценности комманд (каждой, разница, разница/время)

In [ ]:
```
# Считаем сумму золота для команды Radiant
df_test_features_extended['r_total_gold'] = df_test_features_extended[['r1_gold', '

# Считаем сумму золота для команды Dire
df_test_features_extended['d_total_gold'] = df_test_features_extended[['d1_gold', '

# Вычисляем разницу между суммами золота команд
df_test_features_extended['gold_diff'] = df_test_features_extended['r_total_gold']

# Вычисляем разницу между суммами золота команд, деленную на время игры
df_test_features_extended['gold_diff_per_second'] = df_test_features_extended['gold

# Переводим обновленный DataFrame в массив NumPy
X_test = df_test_features_extended.values
```

```
# Считаем сумму золота для команды Radiant
df_train_features_extended['r_total_gold'] = df_train_features_extended[['r1_gold',

# Считаем сумму золота для команды Dire
df_train_features_extended['d_total_gold'] = df_train_features_extended[['d1_gold',

# Вычисляем разницу между суммами золота команд
df_train_features_extended['gold_diff'] = df_train_features_extended['r_total_gold'

# Вычисляем разницу между суммами золота команд, деленную на время игры
df_train_features_extended['gold_diff_per_second'] = df_train_features_extended['go
```

И попытался подобрать гиперпараметры, но это не улучшило результат

In [ ]:
```
# Определяем модель CatBoost
cb_model = CatBoostClassifier(verbose=False)

# Задаем сетку параметров, которые хотим проверить
param_grid = {
    'depth': [5, 7, 9],
    'iterations': [50, 100],
    'learning_rate': [0.01, 0.05, 0.1],
    'l2_leaf_reg': [1, 3, 5],
    'border_count': [32, 64, 128]
}


# Создаем объект GridSearchCV
grid_search = GridSearchCV(estimator=cb_model, param_grid=param_grid, cv=3, scoring

# Выполняем решетчатый поиск на обучающих данных
grid_search.fit(X_train, y_train)

# Выводим наилучшие параметры
print("Наилучшие параметры: ", grid_search.best_params_)
```

Итог

Лучшие тестовые показатели были у LGBM

Но лучший показатель при submit оказался у CatBoost