Ансамбли моделей машинного обучения. Часть 1.

Подготовка

Загрузка датасета

In [ ]:
```
import pandas as pd
df = pd.read_csv('GM_players_statistics.csv')
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
ID                          int64
name                       object
username                   object
profile                    object
user_id                     int64
title                      object
fide                      float64
country                    object
followers                   int64
joined                     object
last_online                object
current_rapid_rating      float64
highest_rapid_rating      float64
highest_rapid_date         object
rapid_win                 float64
rapid_draw                float64
rapid_loss                float64
current_blitz_rating      float64
highest_blitz_rating      float64
highest_blitz_date         object
blitz_win                 float64
blitz_draw                float64
blitz_loss                float64
current_bullet_rating     float64
highest_bullet_rating     float64
highest_bullet_date        object
bullet_win                float64
bullet_draw               float64
bullet_loss               float64
highest_tactics_rating      int64
highest_tactics_date       object
highest_puzzle_rush_score float64
is_streamer                  bool
status                     object
league                     object
dtype: object
```

In [ ]:
```
df.head()
```

Out[ ]:

	ID	name	username	profile	user_id
0	0	Komodo Engine	komodochess	https://www.chess.com/member/KomodoChess	24944922
1	1	Vojtěch Plát	vojtechplat	https://www.chess.com/member/VojtechPlat	37712368
2	2	PlayMagnus Carlsen	playmagnus	https://www.chess.com/member/PlayMagnus	19578862
3	3	Magnus Carlsen	magnuscarlsen	https://www.chess.com/member/MagnusCarlsen	3889224
4	4	Fabiano Caruana	fabianocaruana	https://www.chess.com/member/FabianoCaruana	11177810

5 rows × 35 columns

Заполнение пропусков

In [ ]:
```
from sklearn.impute import SimpleImputer

num_cols = []

for col in df.columns:
    temp_null_count = df[df[col].isnull()].shape[0]
    dt = str(df[col].dtype)

    if dt=='float64' or dt=='int64':
        num_cols.append(col)

    if temp_null_count == 0:
        continue

    imp_strategy=''
    if dt=='float64' or dt=='int64':
        imp_strategy = 'mean'
    else:
        imp_strategy = 'most_frequent'

    imp_num = SimpleImputer(strategy=imp_strategy)
    data_num_imp = imp_num.fit_transform(df[[col]])
    df[[col]] = data_num_imp

df[num_cols]
```

Out[ ]:

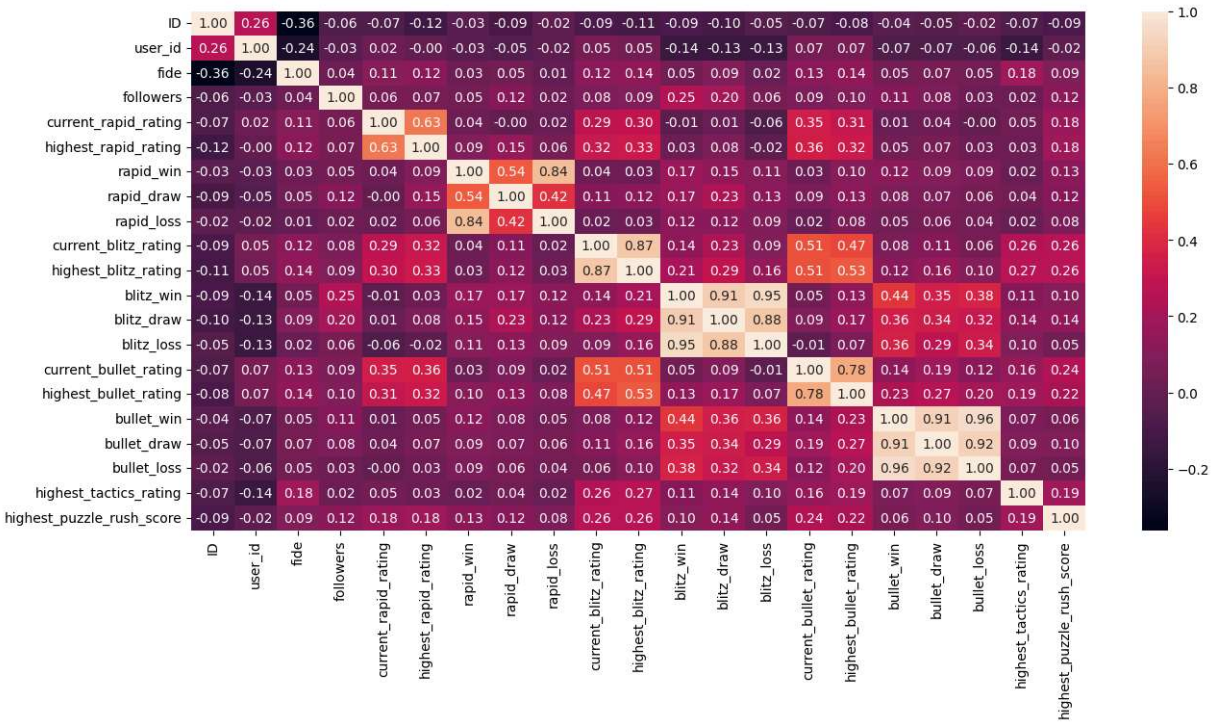
	ID	user_id	fide	followers	current_rapid_rating	highest_rapid_rating
0	0	24944922	3411.000000	494	2382.517536	2555.735206
1	1	37712368	2950.000000	360	2383.000000	2672.000000
2	2	19578862	2882.000000	199	2382.517536	2555.735206
3	3	3889224	2882.000000	142482	2925.000000	2977.000000
4	4	11177810	2835.000000	14445	2794.000000	3045.000000
...	...	...	...	...	...	...
1503	1503	13526852	2002.937568	2	2382.517536	2555.735206
1504	1504	4042329	2002.937568	127	2375.000000	2555.000000
1505	1505	83663714	2002.937568	37	2382.517536	2555.735206
1506	1506	148025331	2002.937568	5	2382.517536	2555.735206
1507	1507	32236996	2002.937568	884	2346.000000	2550.000000

1508 rows × 21 columns

Матрица корреляций

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots(figsize=(15, 7))
sns.heatmap(df[num_cols].corr(method='pearson'), ax=ax, annot=True, fmt='.2f')
```

Out[ ]: <Axes: >



Кодирование категориального признака `league`

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

cat_enc_le = le.fit_transform(df['league'])
df['league_coded'] = cat_enc_le
```

Разделение выборки на обучающую и тестовую

In [ ]:
```
from sklearn.model_selection import train_test_split

x_df = df[num_cols] # числовые колонки

y_df = df[['league_coded']] # колонка с закодированной лигой

x_train, x_test, y_train, y_test = train_test_split(x_df, y_df, test_size=0.25, ran
```

Бэггинг

In [ ]:
```
from sklearn.ensemble import BaggingClassifier

bc = BaggingClassifier(n_estimators=5, oob_score=True, random_state=42)
bc.fit(x_df, y_df)
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:802: DataConversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:789: UserWarning: Some inputs do not have OOB scores. This probably means too few
estimators were used to compute any reliable oob estimates.
  warn(
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:795: RuntimeWarning: invalid value encountered in divide
  oob_decision_function = predictions / predictions.sum(axis=1)[:, np.newaxis]
```

Out[ ]:

BaggingClassifier

```
BaggingClassifier(n_estimators=5, oob_score=True, random_state=42)
```

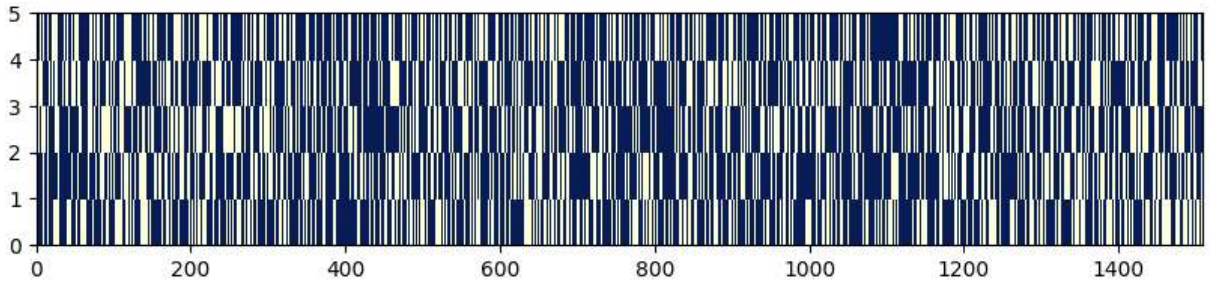
In [ ]:
```
bc.estimators_samples_
```

Out[ ]:
```
[array([ 475, 1065,  970, ..., 1255,  540,   67]),
 array([ 686,  663, 1011, ..., 1338, 1336,  384]),
 array([1366,  430,  800, ...,  740,   37, 1186]),
 array([ 800,  581,   63, ..., 1354,  408,   73]),
 array([ 350,   25, 1254, ...,  930,  264,   97])]
```

In [ ]:
```
import numpy as np

# Сконвертируем эти данные в двоичную матрицу,
# 1 соответствует элементам, попавшим в обучающую выборку
bin_array = np.zeros((5, x_df.shape[0]))
for i in range(5):
    for j in bc.estimators_samples_[i]:
        bin_array[i][j] = 1
bin_array
```

Out[ ]:
```
array([[1., 1., 0., ..., 1., 1., 0.],
       [1., 1., 0., ..., 1., 1., 0.],
       [1., 1., 0., ..., 1., 1., 1.],
       [1., 0., 0., ..., 1., 0., 0.],
       [1., 1., 1., ..., 1., 0., 1.]])
```

In [ ]:
```
import matplotlib.pyplot as plt

# И визуализируем (синим цветом показаны данные, которые попали в обучающую выборку
fig, ax = plt.subplots(figsize=(10,2))
ax.pcolor(bin_array, cmap='YlGnBu')
plt.show()
```



In [ ]:
```
# Оценим Out-of-bag error, теоретическое значение 37%
for i in range(5):
    cur_data = bin_array[i]
    len_cur_data = len(cur_data)
    sum_cur_data = sum(cur_data)
    (len(bin_array[0]) - sum(bin_array[0])) / len(bin_array[0])
    oob_i = (len_cur_data - sum_cur_data) / len_cur_data
    print('Для модели № {} размер OOB составляет {}%'.format(i+1, round(oob_i, 4)*1
```

```
Для модели № 1 размер OOB составляет 36.54%
Для модели № 2 размер OOB составляет 36.67%
Для модели № 3 размер OOB составляет 36.07%
Для модели № 4 размер OOB составляет 37.07%
Для модели № 5 размер OOB составляет 37.8%
```

In [ ]:
```
# Out-of-bag error, возвращаемый классификатором
# Для классификации используется метрика accuracy
bc.oob_score_, 1-bc.oob_score_
```

Out[ ]:
```
(0.30238726790450926, 0.6976127320954908)
```

In [ ]:
```
# Параметр oob_decision_function_ возвращает вероятности
# принадлежности объекта к классам на основе oob
# В данном примере три класса,
# значения пап могут возвращаться в случае маленькой выборки
bc.oob_decision_function_[55:70]
```

Out[ ]:
```
array([[0.33333333, 0.        , 0.        , 0.        , 0.        ,
        0.33333333, 0.        , 0.33333333],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.33333333, 0.        , 0.33333333, 0.        ,
        0.33333333, 0.        , 0.        ],
       [0.        , 0.        , 0.5       , 0.        , 0.        ,
        0.5       , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.5       , 0.        , 0.5       ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.66666667, 0.33333333, 0.        ],
       [       nan,        nan,        nan,        nan,        nan,
               nan,        nan,        nan],
       [0.        , 0.        , 0.        , 0.5       , 0.        ,
        0.5       , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.33333333, 0.33333333, 0.        , 0.        ,
        0.33333333, 0.        , 0.        ]])
```

In [ ]:
```
from IPython.display import Image
from io import StringIO
from sklearn.tree import export_graphviz
import pydotplus

# Визуализация дерева
def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_name
                    filled=True, rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()
```

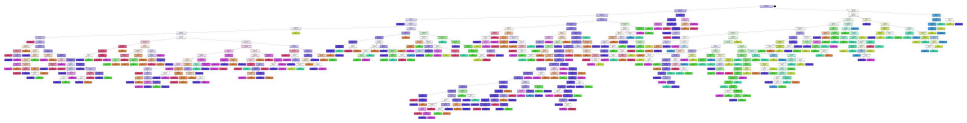
In [ ]:
```
Image(get_png_tree(bc.estimators_[0], x_df.columns), width='80%')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.842621 to fit
```

Out[ ]:


In [ ]:
```
Image(get_png_tree(bc.estimators_[1], x_df.columns), width='80%')
```

Out[ ]:



In [ ]:
```
Image(get_png_tree(bc.estimators_[2], x_df.columns), width='80%')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.954499 to fit
```

Out[ ]:

In [ ]:
```
Image(get_png_tree(bc.estimators_[3], x_df.columns), width='80%')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.944648 to fit
```

Out[ ]:

In [ ]:
```
Image(get_png_tree(bc.estimators_[4], x_df.columns), width='80%')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.942637 to fit
```

Out[ ]:

In [ ]:
```
def make_meshgrid(x, y, h=1):
    """Create a mesh of points to plot in

    Parameters
    ----------
    x: data to base x-axis meshgrid on
    y: data to base y-axis meshgrid on
    h: stepsize for meshgrid, optional

    Returns
    -------
    xx, yy : ndarray
    """
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    """Plot the decision boundaries for a classifier.

    Parameters
    ----------
    ax: matplotlib axes object
    clf: a classifier
    xx: meshgrid ndarray
    yy: meshgrid ndarray
    params: dictionary of params to pass to contourf, optional
    """
```

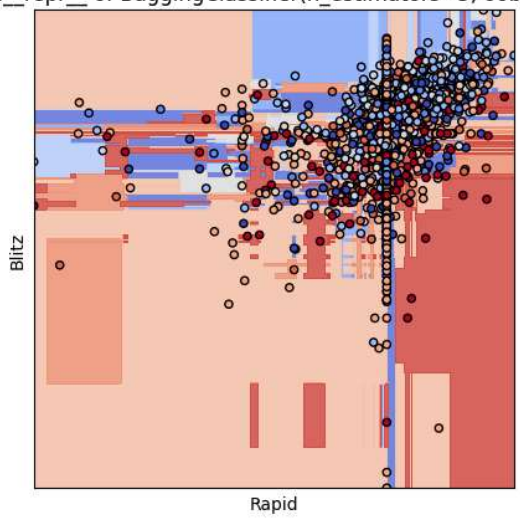
```
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    #Можно проверить все ли метки классов предсказываются
    #print(np.unique(Z))
    out = ax.contourf(xx, yy, Z, **params)
    return out


def plot_cl(clf):
    title = clf.__repr__
    clf.fit(x_df[['highest_rapid_rating', 'highest_blitz_rating']].values, y_df.val
    fig, ax = plt.subplots(figsize=(5,5))
    X0, X1 = x_df[['highest_rapid_rating']].values, x_df[['highest_blitz_rating']].
    xx, yy = make_meshgrid(X0, X1)
    plot_contours(ax, clf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
    ax.scatter(X0, X1, c=y_df.values, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xlabel('Rapid')
    ax.set_ylabel('Blitz')
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)
    plt.show()
```

In [ ]:

```
from sklearn.tree import DecisionTreeClassifier

plot_cl(BaggingClassifier(n_estimators=5, oob_score=True, random_state=42))
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:802: DataConversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:789: UserWarning: Some inputs do not have OOB scores. This probably means too few
estimators were used to compute any reliable oob estimates.
  warn(
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_bagging.p
y:795: RuntimeWarning: invalid value encountered in divide
  oob_decision_function = predictions / predictions.sum(axis=1)[:, np.newaxis]
3045.0 3332.0
```

```
<bound method BaseEstimator.__repr__ of BaggingClassifier(n_estimators=5, oob_score=True, random_state=42)>
```



Случайный лес

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=5, oob_score=True, random_state=10)
rfc.fit(x_df, y_df)
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:1152: 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().
  return fit_method(estimator, *args, **kwargs)
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_forest.py:578: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
  warn(
```

Out[ ]:
```
RandomForestClassifier(n_estimators=5, oob_score=True, random_state=10)
```

In [ ]:
```
rfc.oob_score_, 1 - rfc.oob_score_
```

Out[ ]:
```
(0.29509283819628646, 0.7049071618037135)
```

In [ ]:
```
rfc.oob_decision_function_[55:70]
```

Out[ ]:
```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.33333333, 0.        , 0.        , 0.33333333, 0.        ,
        0.33333333, 0.        , 0.        ],
       [1.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.33333333, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.33333333, 0.33333333],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.5       , 0.5       , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.5       , 0.5       , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.33333333, 0.        , 0.        , 0.        , 0.        ,
        0.66666667, 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 1.        ],
       [0.        , 0.        , 0.25      , 0.25      , 0.        ,
        0.5       , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.5       , 0.        , 0.        , 0.5       ,
        0.        , 0.        , 0.        ]])
```

In [ ]:
```
Image(get_png_tree(rfc.estimators_[0], x_df.columns), width="500")
```

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.727881 to fit

Out[ ]:


In [ ]:
```
Image(get_png_tree(rfc.estimators_[1], x_df.columns), width="500")
```

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.806652 to fit

Out[ ]:


In [ ]:
```
Image(get_png_tree(rfc.estimators_[2], x_df.columns), width="500")
```

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.676627 to fit

Out[ ]:


In [ ]:
```
Image(get_png_tree(rfc.estimators_[3], x_df.columns), width="500")
```

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.765888 to fit

Out[ ]:

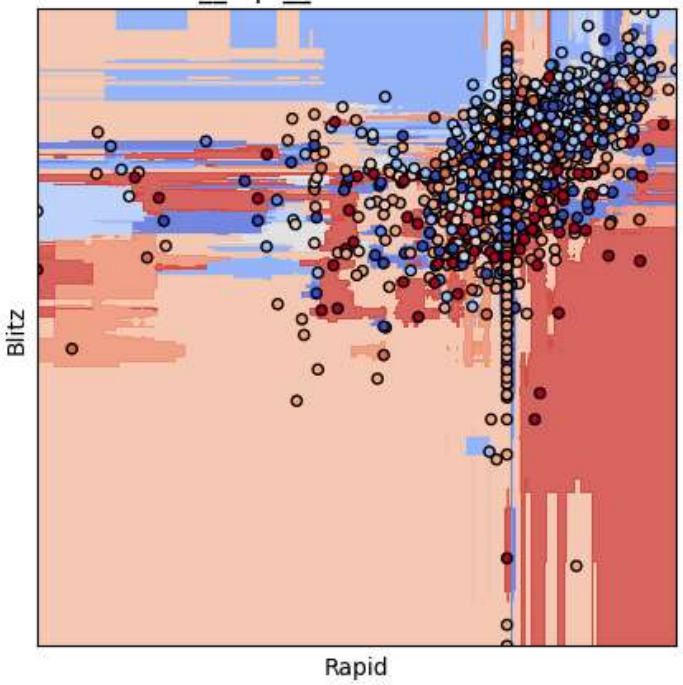
In [ ]:
```
Image(get_png_tree(rfc.estimators_[4], x_df.columns), width="500")
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.88581 to fit
```

Out[ ]:

In [ ]:
```
plot_cl(RandomForestClassifier(random_state=1))
```

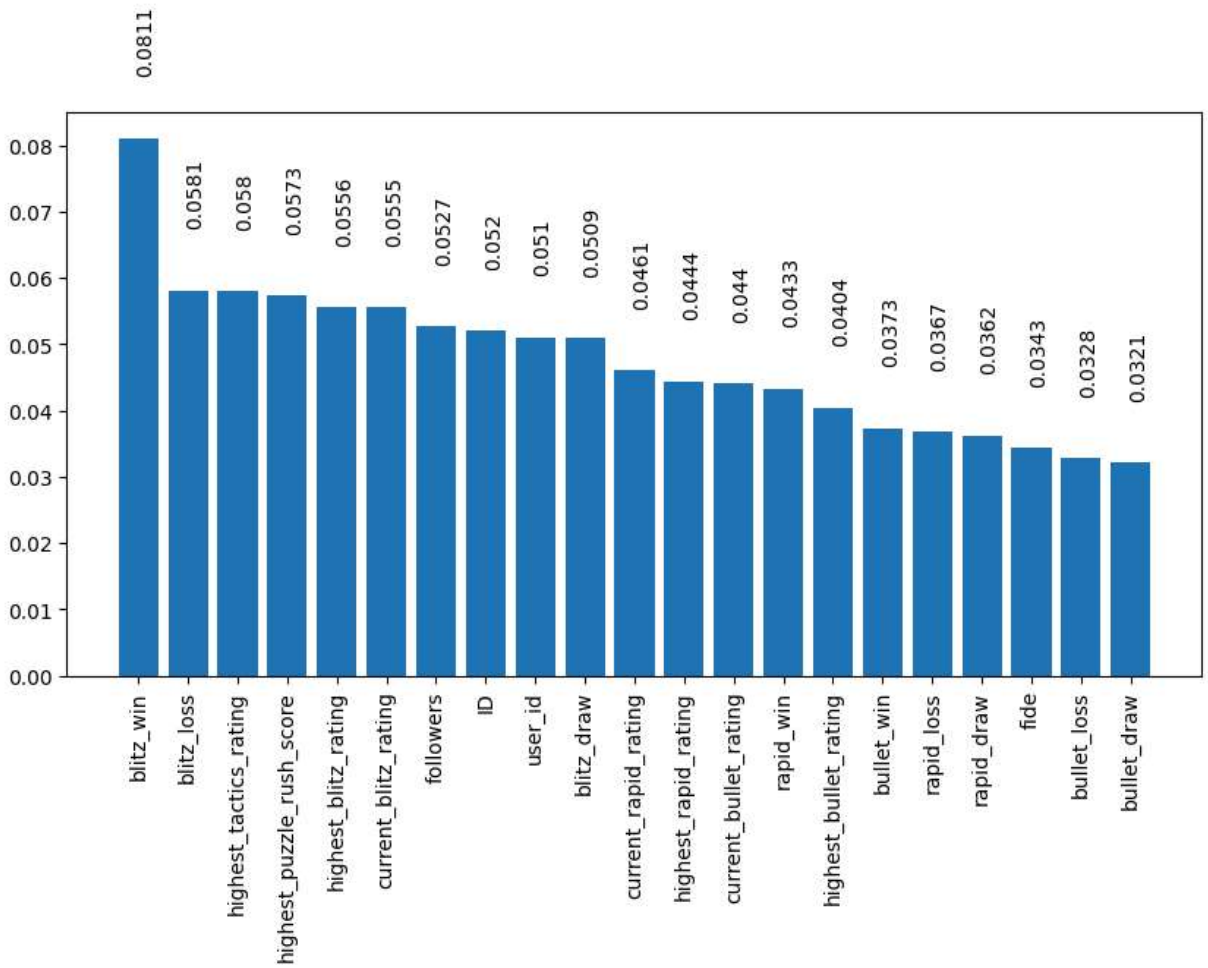
```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:1152: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
3045.0 3332.0
<bound method BaseEstimator.__repr__ of RandomForestClassifier(random_state=1)>
```



In [ ]:
```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(10,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importance
    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,4)), rotation=90)
    plt.show()
    return labels, data
```

In [ ]:
```
# Важность признаков
_,_ = draw_feature_importances(rfc, x_df)
```



AdaBoost

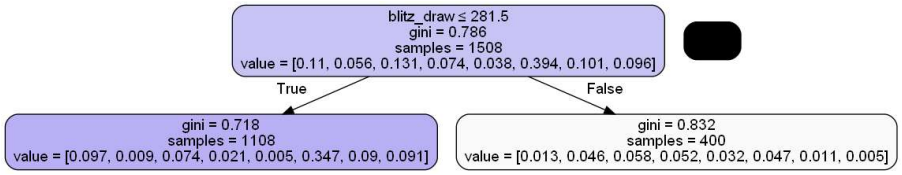
In [ ]:
```
from sklearn.ensemble import AdaBoostClassifier

abc = AdaBoostClassifier(n_estimators=5, algorithm='SAMME', random_state=42)
abc.fit(x_df, y_df)
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\utils\validation.p
y:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

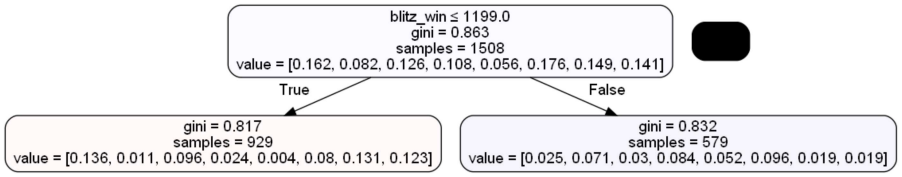
Out[ ]:
AdaBoostClassifier

```
AdaBoostClassifier(algorithm='SAMME', n_estimators=5, random_state=42)
```

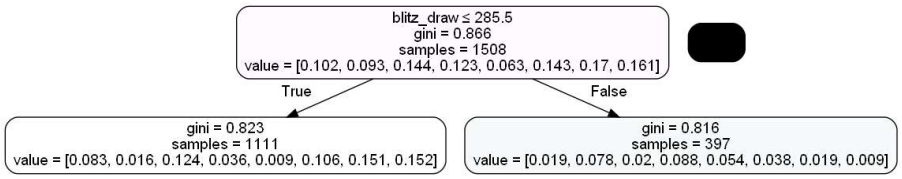
In [ ]:
```
Image(get_png_tree(abc.estimators_[0], x_df.columns), width='75%')
```

Out[ ]:


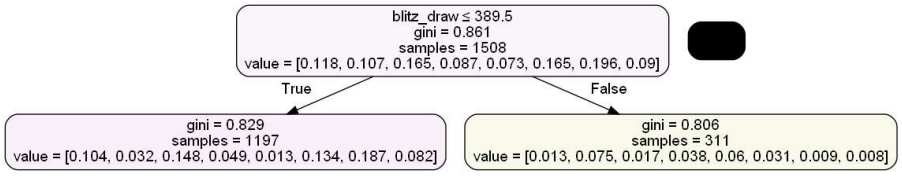
In [ ]:
```
Image(get_png_tree(abc.estimators_[1], x_df.columns), width='75%')
```

Out[ ]:


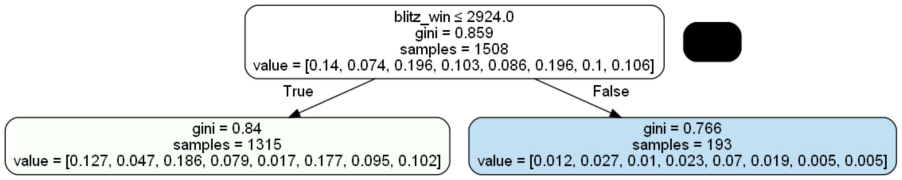
In [ ]:
```
Image(get_png_tree(abc.estimators_[2], x_df.columns), width='75%')
```

Out[ ]:


In [ ]:
```
Image(get_png_tree(abc.estimators_[3], x_df.columns), width='75%')
```

Out[ ]:


In [ ]:
```
Image(get_png_tree(abc.estimators_[4], x_df.columns), width='75%')
```

Out[ ]:


In [ ]:
```
abc.estimator_weights_
```

Out[ ]:
```
array([1.55919904, 0.75045841, 0.79257152, 0.91067985, 0.8765446 ])
```

In [ ]:
```
df1 = abc.decision_function(x_df)
df1.shape
```

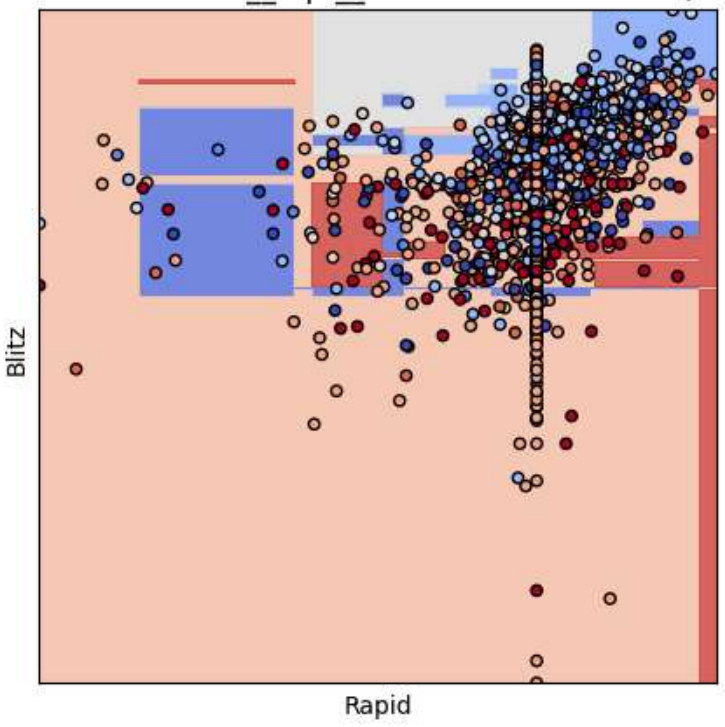
Out[ ]:
```
(1508, 8)
```

In [ ]:
```
df1[:10]
```

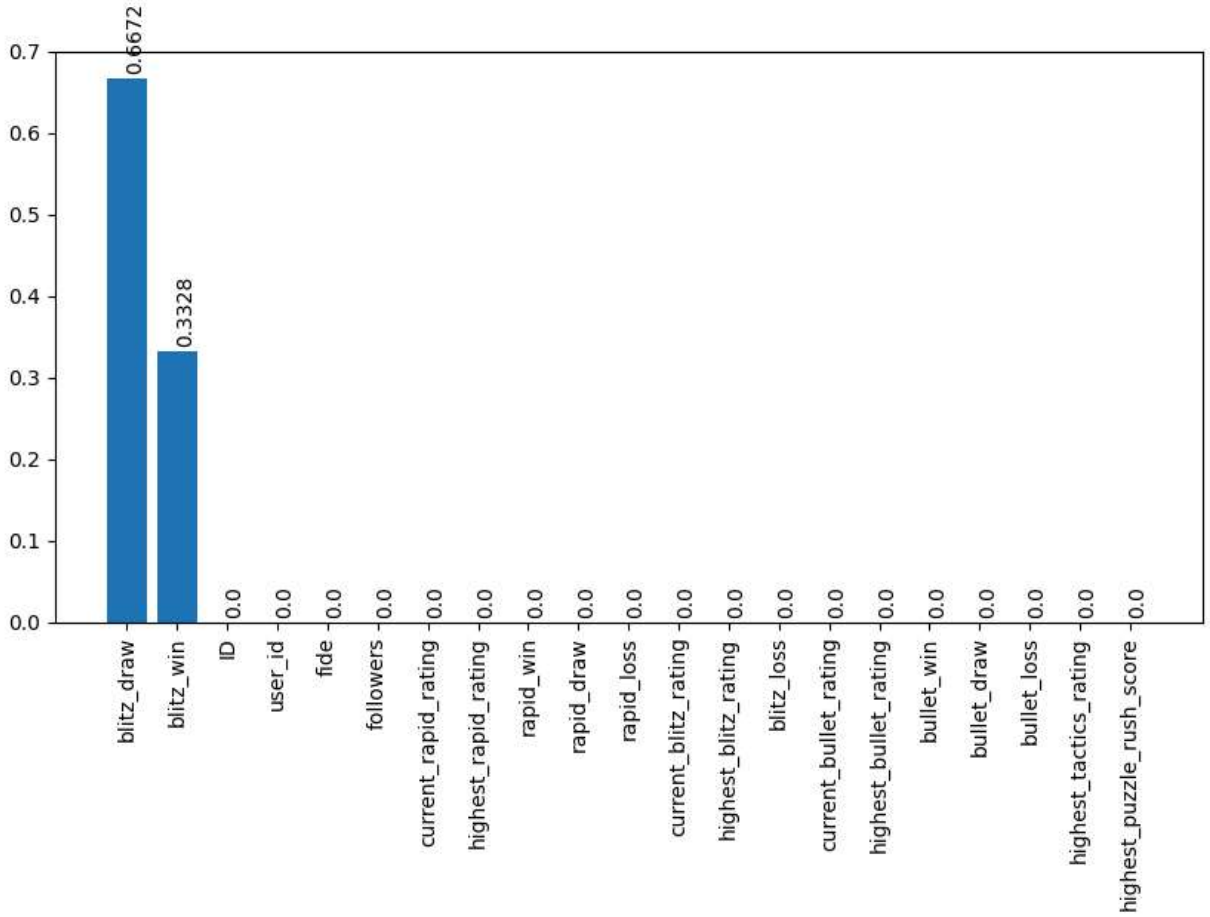
Out[ ]:
```
array([[-0.14285714, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.39700044,  0.07000448,  0.04239792],
       [ 0.03255444, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.22158886,  0.07000448,  0.04239792],
       [-0.14285714, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.39700044,  0.07000448,  0.04239792],
       [-0.14285714, -0.14285714,  0.42647173,  0.04239792, -0.14285714,
         0.03255444,  0.07000448, -0.14285714],
       [ 0.03255444, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.22158886,  0.07000448,  0.04239792],
       [ 0.03255444, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.22158886,  0.07000448,  0.04239792],
       [-0.14285714,  0.07000448,  0.22158886,  0.04239792,  0.06202573,
         0.03255444, -0.14285714, -0.14285714],
       [-0.14285714, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.39700044,  0.07000448,  0.04239792],
       [-0.14285714, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.39700044,  0.07000448,  0.04239792],
       [ 0.03255444, -0.14285714,  0.06202573, -0.14285714, -0.14285714,
         0.22158886,  0.07000448,  0.04239792]])
```

In [ ]:
```
plot_cl(AdaBoostClassifier(random_state=1))
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\utils\validation.py:1183: 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)
3045.0 3332.0
<bound method BaseEstimator.__repr__ of AdaBoostClassifier(random_state=1)>
```



In [ ]:
```
# Важность признаков
_,_ = draw_feature_importances(abc, x_df)
```



Градиентный бустинг

In [ ]:
```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(random_state=42)
gbc.fit(x_df, y_df)
```

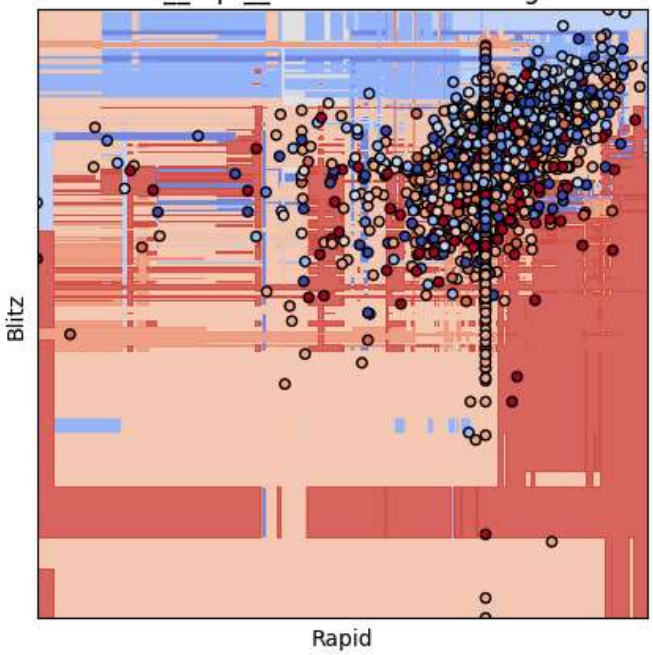
```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_gb.py:42
4: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[ ]:
```
GradientBoostingClassifier(random_state=42)
```

In [ ]:
```
plot_cl(GradientBoostingClassifier(random_state=42))
```

```
C:\Users\Dmitriy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
fra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\ensemble\_gb.py:42
4: 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)
3045.0 3332.0
```

<bound method BaseEstimator.__repr__ of GradientBoostingClassifier(random_state=42)>



In [ ]:
```
# Важность признаков
_,_ = draw_feature_importances(gbc, x_df)
```