РК №2

Фонин Максим Алексеевич ИУ5-61Б

Вариант 17

Задание.

Для заданного набора данных постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы:

- линейная/логистическая регрессия
- случайный лес

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.

Данные: https://www.kaggle.com/mathan/fifa-2018-match-statistics

Импорт библиотек и загрузка датасета

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc, confusion_matrix, precision_score, reca
```

In [2]:
```
df = pd.read_csv('FIFA 2018 Statistics.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Date	Team	Opponent	Goal Scored	Ball Possession %	Attempts	On-Target	Off-Target	Blocked	Cor
0	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	
1	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	
2	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	
3	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	
4	15-06-2018	Morocco	Iran	0	64	13	3	6	4	

5 rows × 27 columns

Обзор датасета

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 27 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Date                    128 non-null    object 
 1   Team                    128 non-null    object 
 2   Opponent                128 non-null    object 
 3   Goal Scored             128 non-null    int64  
 4   Ball Possession %       128 non-null    int64  
 5   Attempts                128 non-null    int64  
 6   On-Target               128 non-null    int64  
 7   Off-Target              128 non-null    int64  
 8   Blocked                 128 non-null    int64  
 9   Corners                 128 non-null    int64  
 10  Offsides                128 non-null    int64  
 11  Free Kicks              128 non-null    int64  
 12  Saves                   128 non-null    int64  
 13  Pass Accuracy %         128 non-null    int64  
 14  Passes                  128 non-null    int64  
 15  Distance Covered (Kms)  128 non-null    int64  
 16  Fouls Committed         128 non-null    int64  
 17  Yellow Card             128 non-null    int64  
 18  Yellow & Red            128 non-null    int64  
 19  Red                     128 non-null    int64  
 20  Man of the Match        128 non-null    object 
 21  1st Goal                94 non-null     float64
 22  Round                   128 non-null    object 
 23  PSO                     128 non-null    object 
 24  Goals in PSO            128 non-null    int64  
 25  Own goals               12 non-null     float64
 26  Own goal Time           12 non-null     float64
dtypes: float64(3), int64(18), object(6)
memory usage: 27.1+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Goal Scored	Ball Possession %	Attempts	On-Target	Off-Target	Blocked	Corners
count	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000
mean	1.320312	49.992188	12.593750	3.914062	5.273438	3.359375	4.718750
std	1.156519	10.444074	5.245827	2.234403	2.409675	2.403195	2.446072
min	0.000000	25.000000	3.000000	0.000000	1.000000	0.000000	0.000000
25%	0.000000	42.000000	9.000000	2.000000	4.000000	1.750000	3.000000
50%	1.000000	50.000000	12.000000	3.500000	5.000000	3.000000	5.000000
75%	2.000000	58.000000	15.000000	5.000000	7.000000	4.000000	6.000000
max	6.000000	75.000000	26.000000	12.000000	11.000000	10.000000	11.000000

8 rows × 21 columns

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Date                          0
Team                          0
Opponent                      0
Goal Scored                   0
Ball Possession %             0
Attempts                      0
On-Target                     0
Off-Target                    0
Blocked                       0
Corners                       0
Offsides                      0
Free Kicks                    0
Saves                         0
Pass Accuracy %               0
Passes                        0
Distance Covered (Kms)        0
Fouls Committed               0
Yellow Card                   0
Yellow & Red                  0
Red                           0
Man of the Match              0
1st Goal                     34
Round                         0
PSO                           0
Goals in PSO                  0
Own goals                   116
Own goal Time               116
dtype: int64
```

In [7]:
```
cols_to_drop = ['Date', '1st Goal', 'Own goals', 'Own goal Time']
```

In [8]:
```
df.drop(cols_to_drop, axis=1, inplace=True)
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Team                       0
Opponent                   0
Goal Scored                0
Ball Possession %          0
Attempts                   0
On-Target                  0
Off-Target                 0
Blocked                    0
Corners                    0
Offsides                   0
Free Kicks                 0
Saves                      0
Pass Accuracy %            0
Passes                     0
Distance Covered (Kms)     0
Fouls Committed            0
Yellow Card                0
Yellow & Red               0
Red                        0
Man of the Match           0
Round                      0
PSO                        0
Goals in PSO               0
dtype: int64
```

In [10]:
```
df[df.duplicated()]
```

Out[10]:

Team	Opponent	Goal Scored	Ball Possession %	Attempts	On-Target	Off-Target	Blocked	Corners	Offsi

0 rows × 23 columns

Дубликаты отсутствуют

Категориальные признаки

In [11]:
```
categorical_columns = df.dtypes[df.dtypes == 'object'].index.tolist()
categorical_columns
```

Out[11]:
```
['Team', 'Opponent', 'Man of the Match', 'Round', 'PSO']
```

In [12]:
```
encoder = OneHotEncoder(sparse=False)
encoded_data = encoder.fit_transform(df[categorical_columns])
encoded_columns = encoder.get_feature_names_out(categorical_columns)
encoded_df = pd.DataFrame(encoded_data, columns=encoded_columns, index=df.index)
df_encoded = pd.concat([df, encoded_df], axis=1)
df_encoded.drop(categorical_columns, axis=1, inplace=True)
```

```
C:\Users\я\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\preprocessing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [13]:
```
df_encoded.head()
```

Out[13]:

	Goal Scored	Ball Possession %	Attempts	On-Target	Off-Target	Blocked	Corners	Offsides	Free Kicks	Saves
0	5	40	13	7	3	3	6	3	11	0
1	0	60	6	0	3	3	2	1	25	2
2	0	43	8	3	3	2	0	1	7	3
3	1	57	14	4	6	4	5	1	13	3
4	0	64	13	3	6	4	5	0	14	2

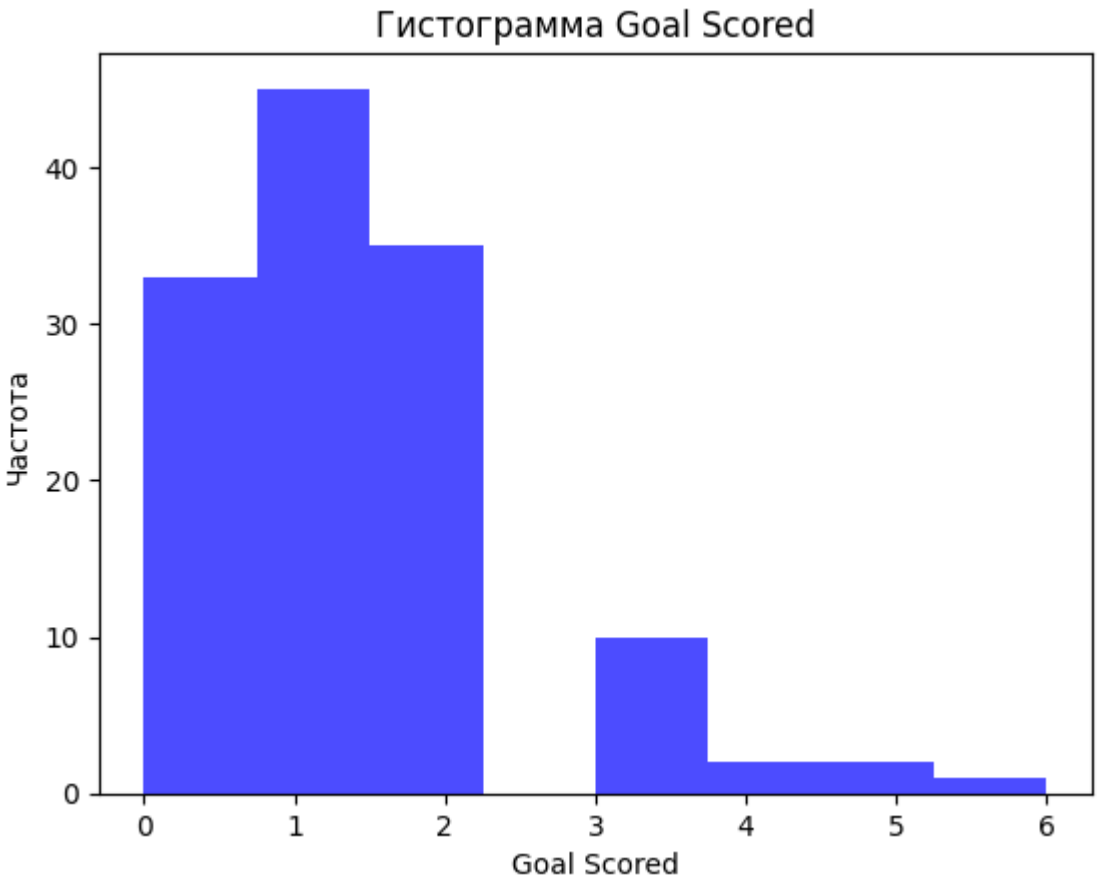
5 rows × 92 columns

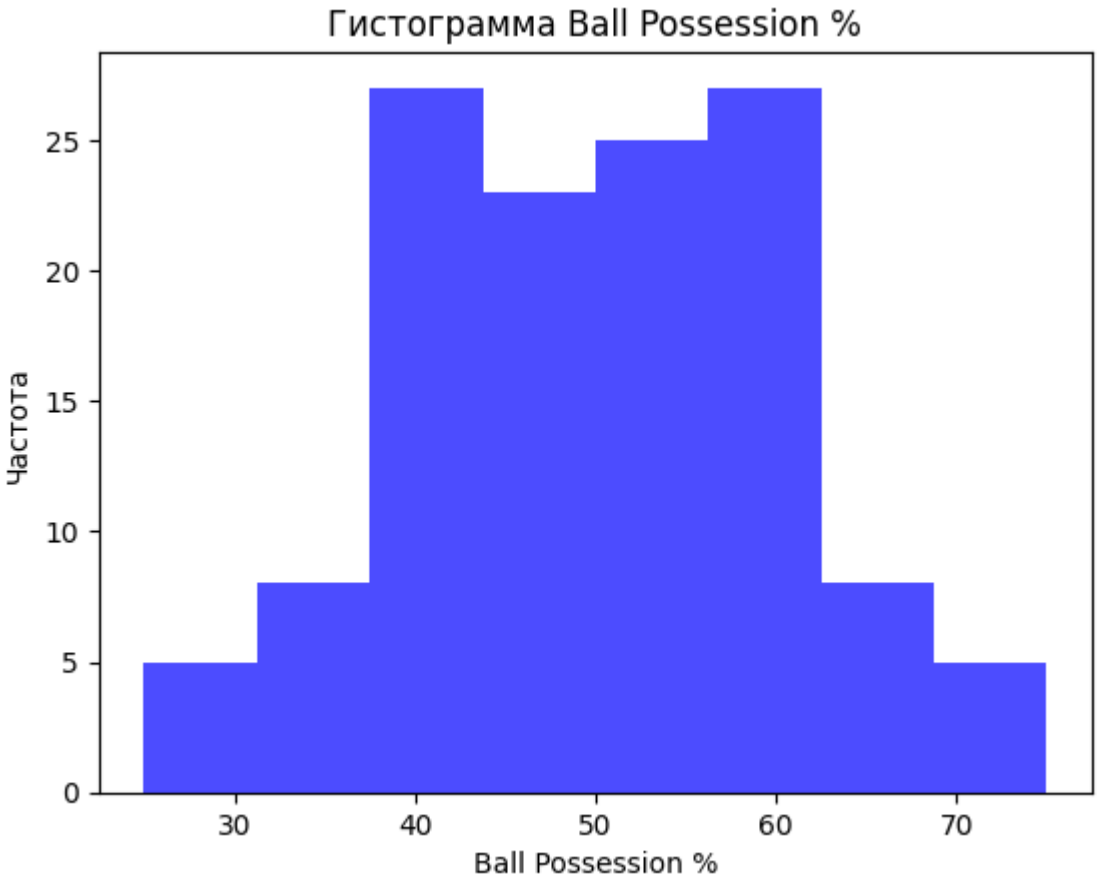
Масштабирование

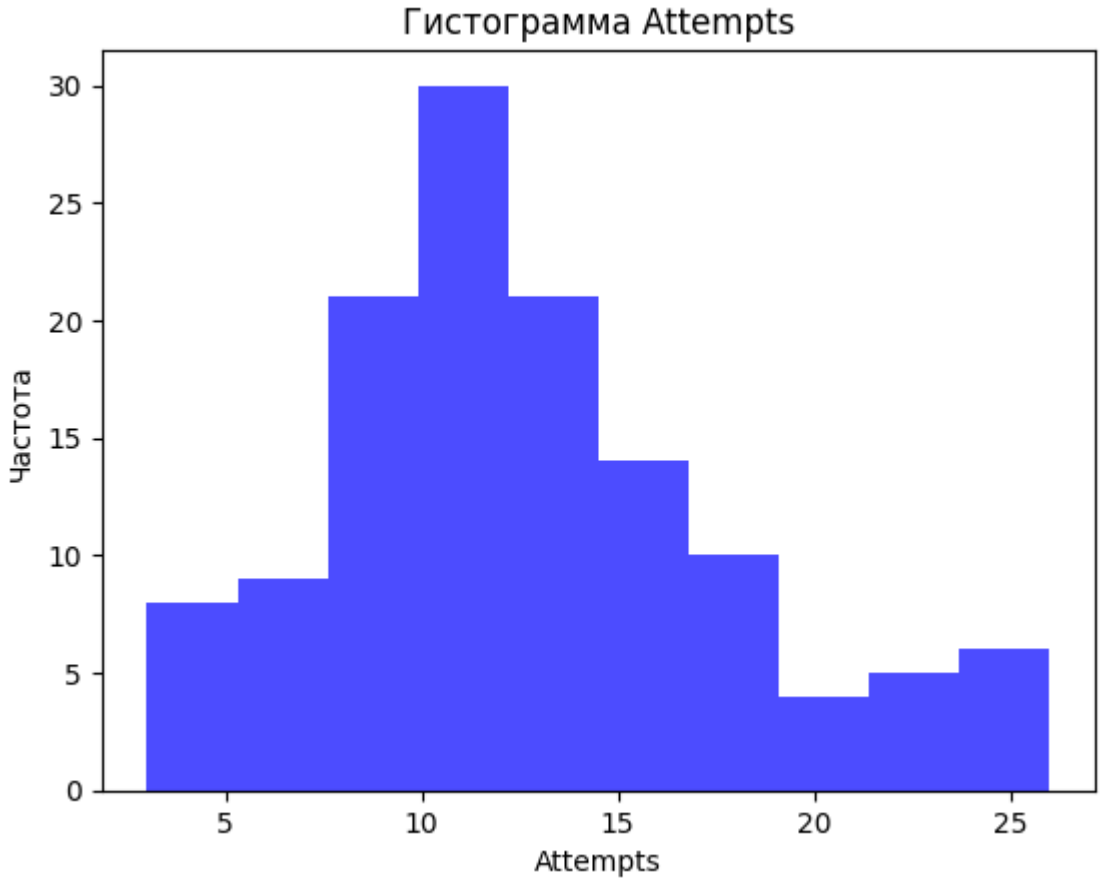
In [14]:
```
numerical_columns = df.dtypes[df.dtypes == 'int64'].index.tolist()
numerical_columns
```

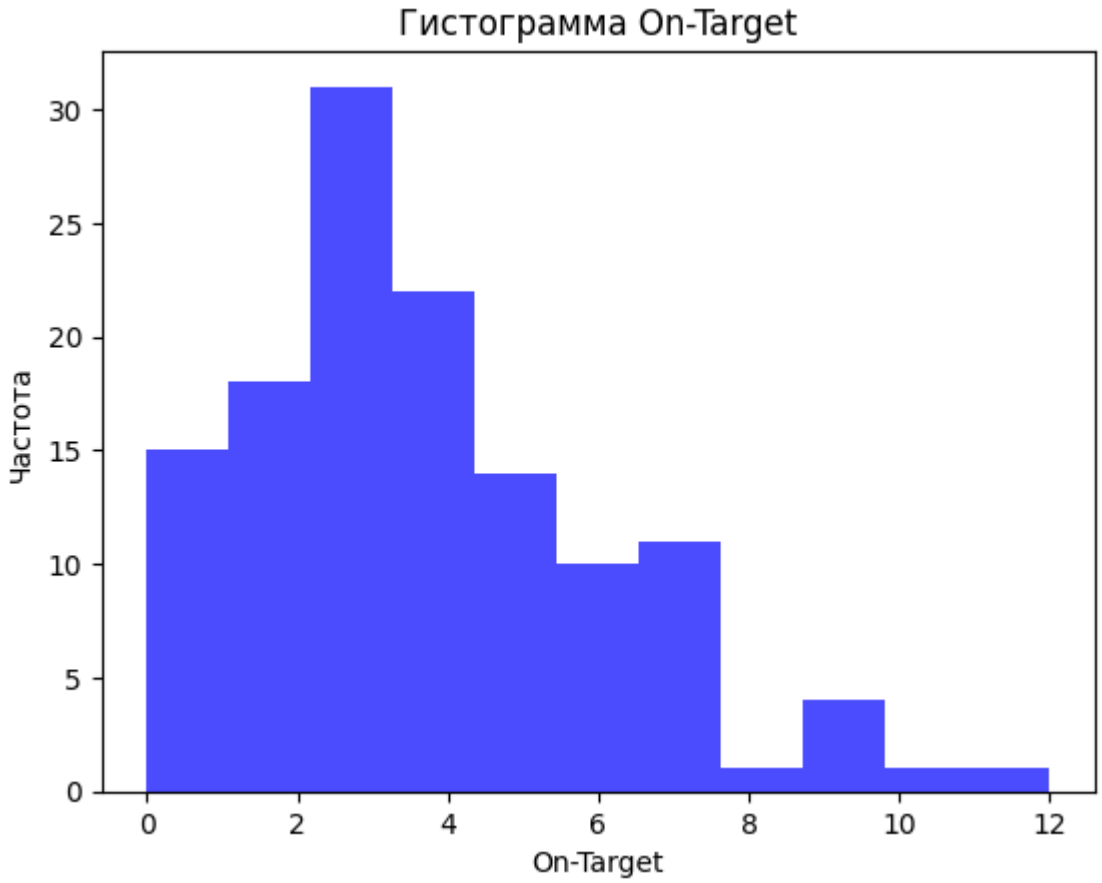
Out[14]:
```
['Goal Scored',
 'Ball Possession %',
 'Attempts',
 'On-Target',
 'Off-Target',
 'Blocked',
 'Corners',
 'Offsides',
 'Free Kicks',
 'Saves',
 'Pass Accuracy %',
 'Passes',
 'Distance Covered (Kms)',
 'Fouls Committed',
 'Yellow Card',
 'Yellow & Red',
 'Red',
 'Goals in PSO']
```

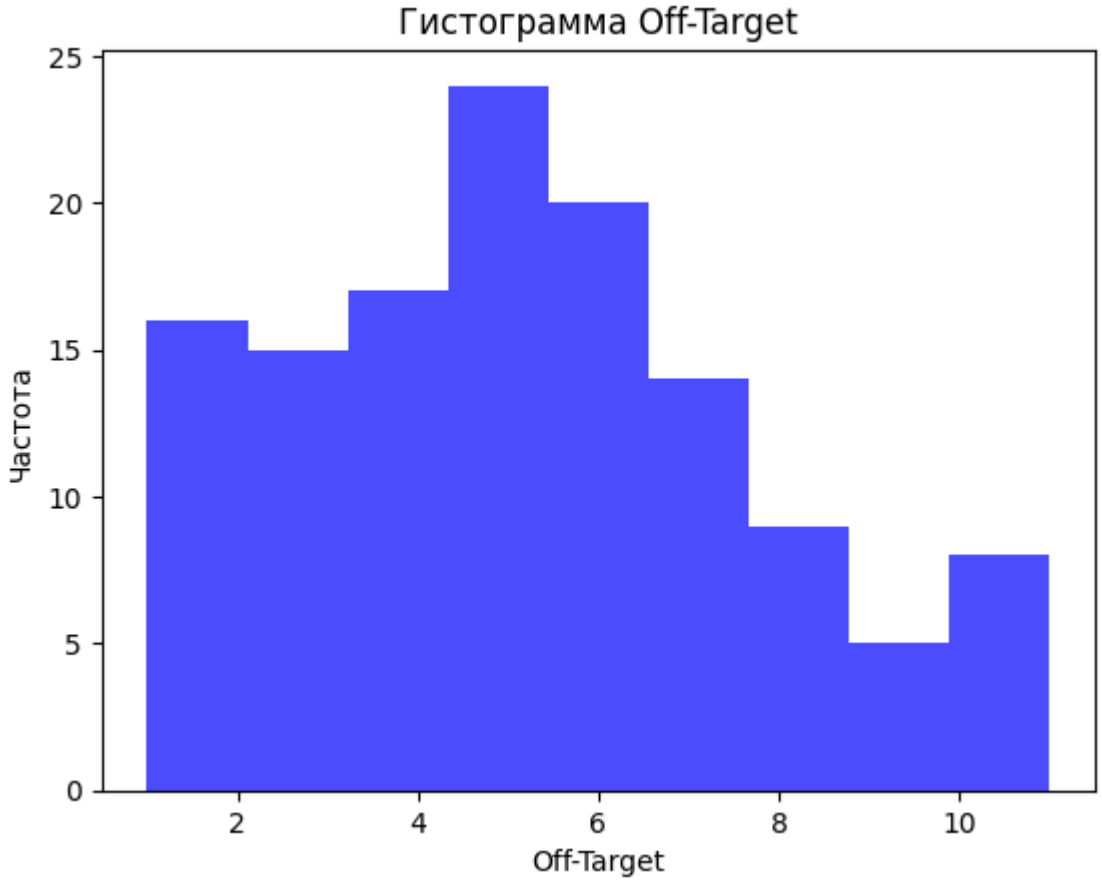
In [15]:
```
for column in numerical_columns:
    plt.figure()
    plt.hist(df_encoded[column], bins='auto', color='blue', alpha=0.7)
    plt.xlabel(column)
    plt.ylabel('Частота')
    plt.title(f'Гистограмма {column}')
    plt.show()
```

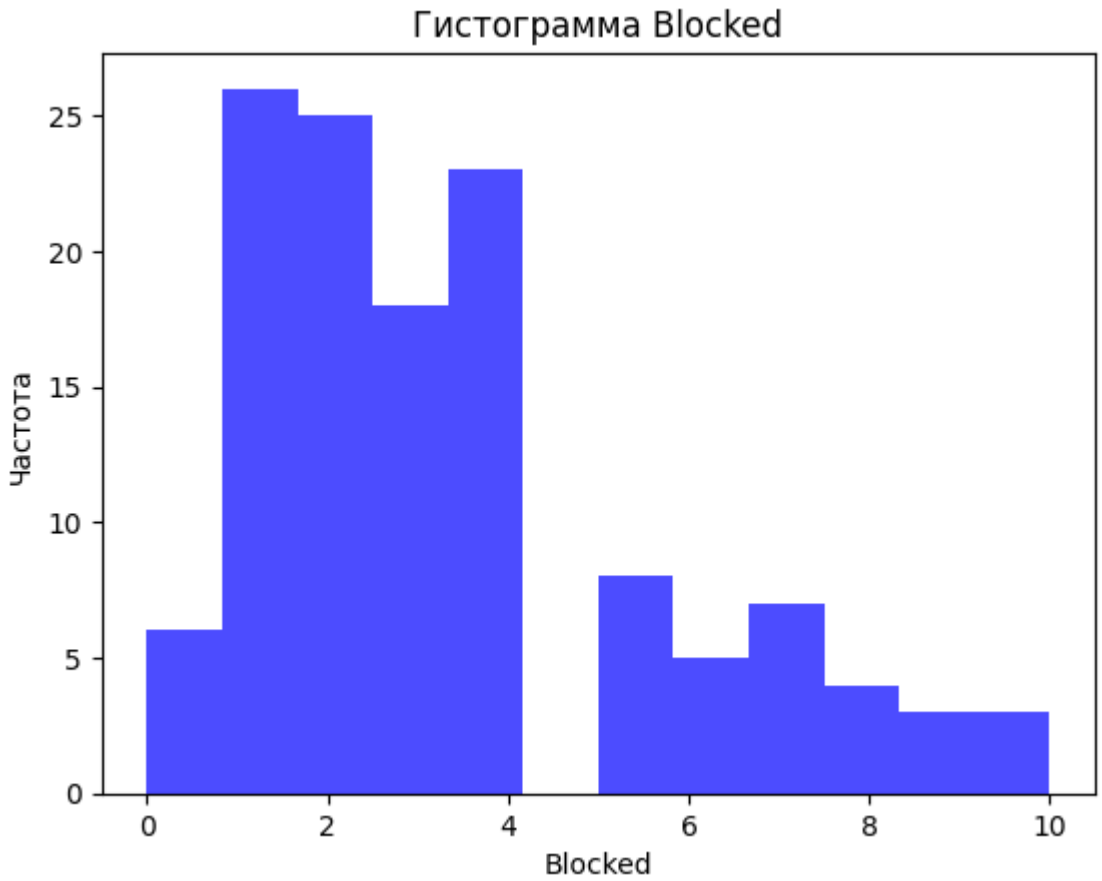

Гистограмма Goal Scored
Частота
0
10
20
30
40
0
1
2
3
4
5
6
Goal Scored

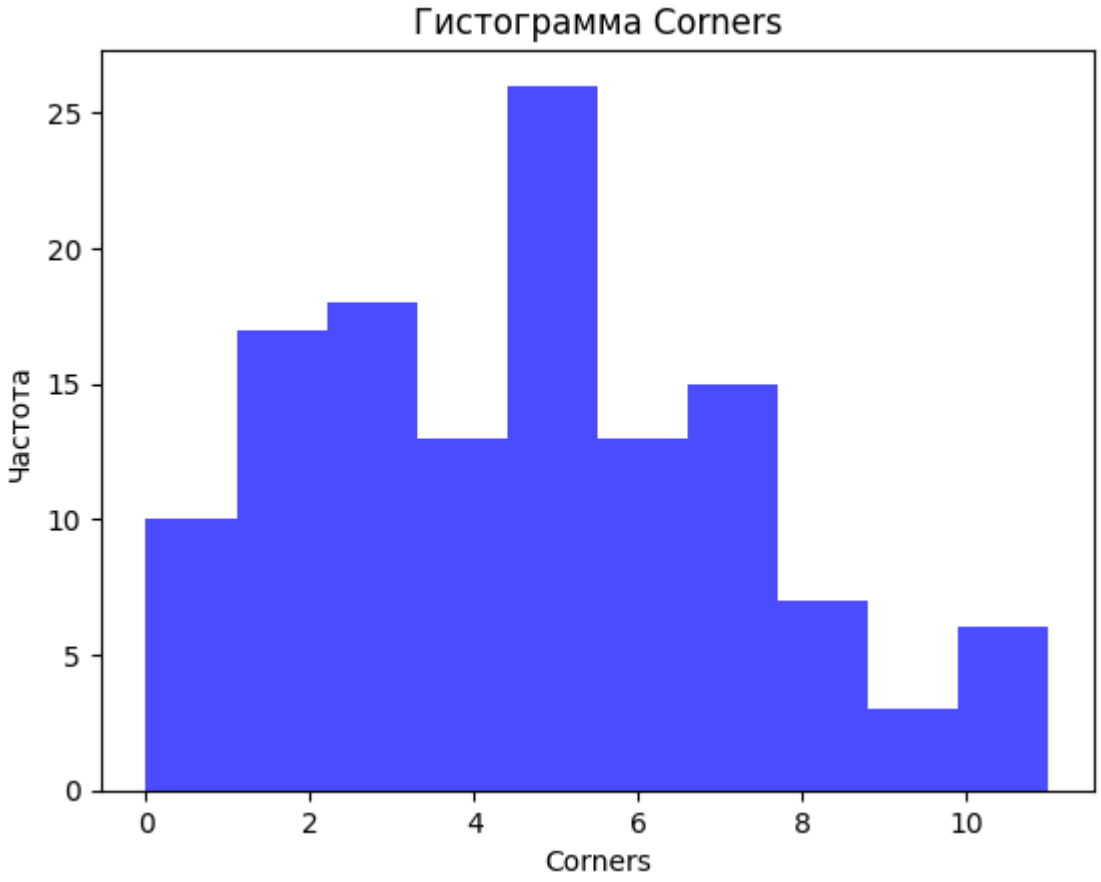

Гистограмма Ball Possession %
Частота
0
5
10
15
20
25
30
40
50
60
70
Ball Possession %

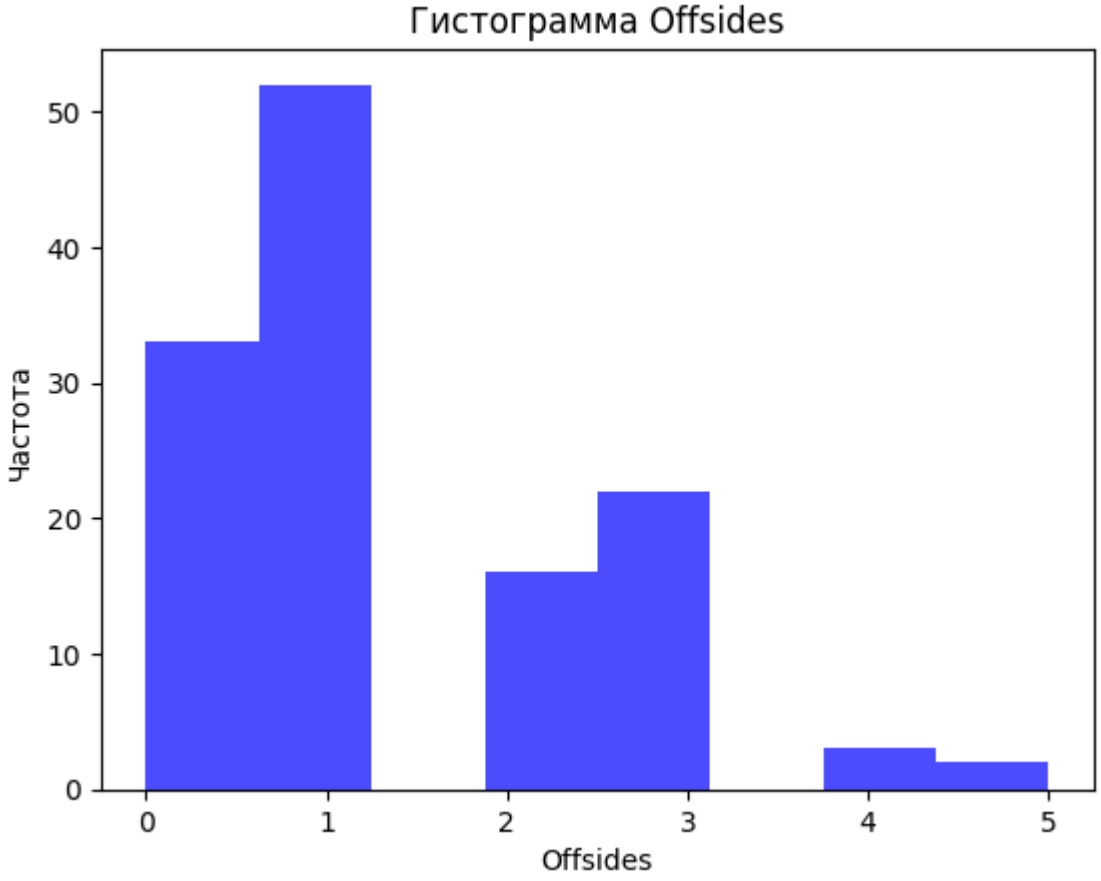

Гистограмма Attempts
Частота
Attempts
0
5
10
15
20
25
30

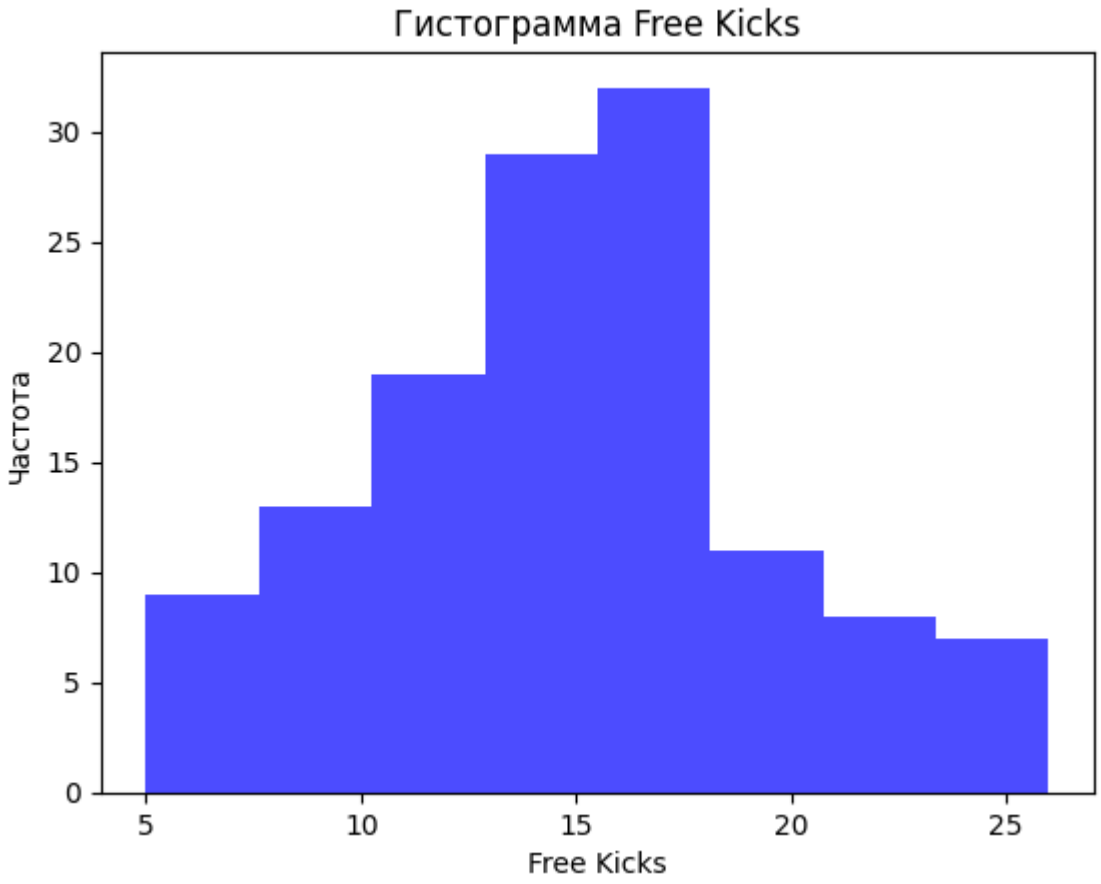

Гистограмма On-Target
Частота
On-Target
0
2
4
6
8
10
12
5
15
20
25
30

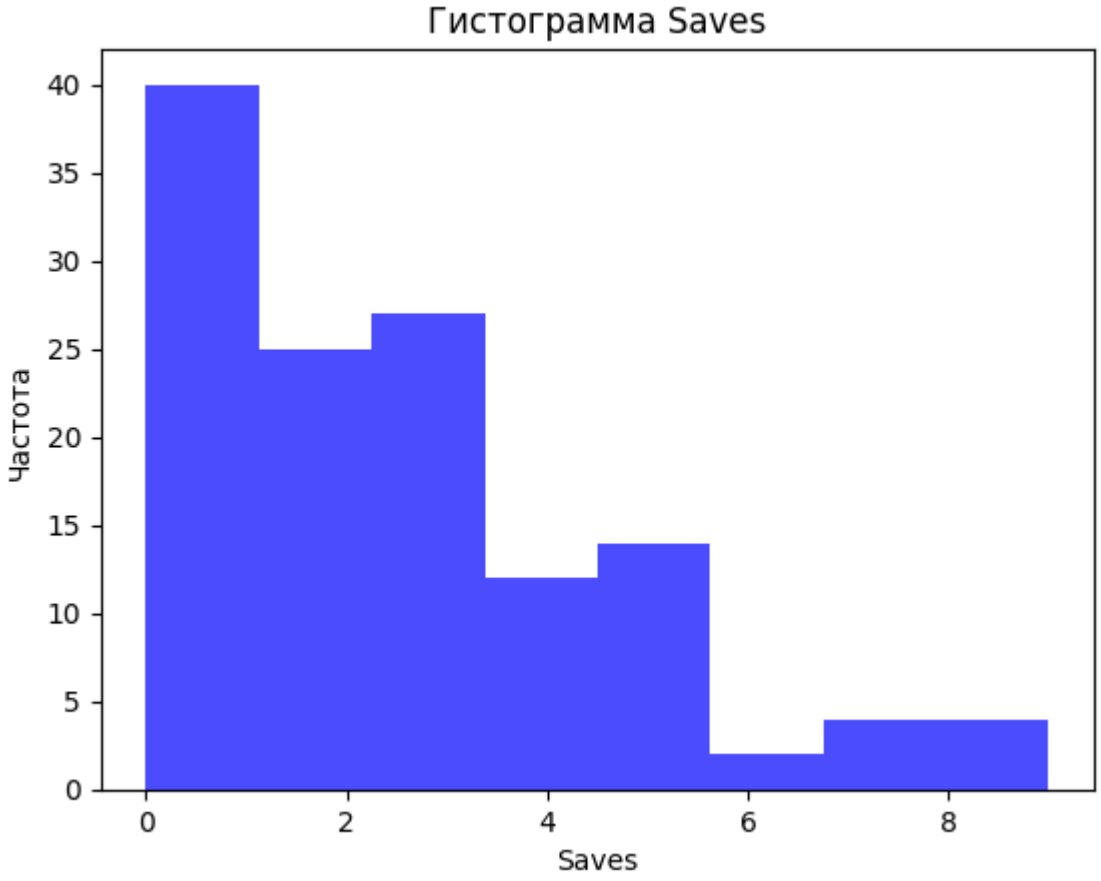

Гистограмма Off-Target
Частота
Off-Target
0
5
10
15
20
25
2
4
6
8
10

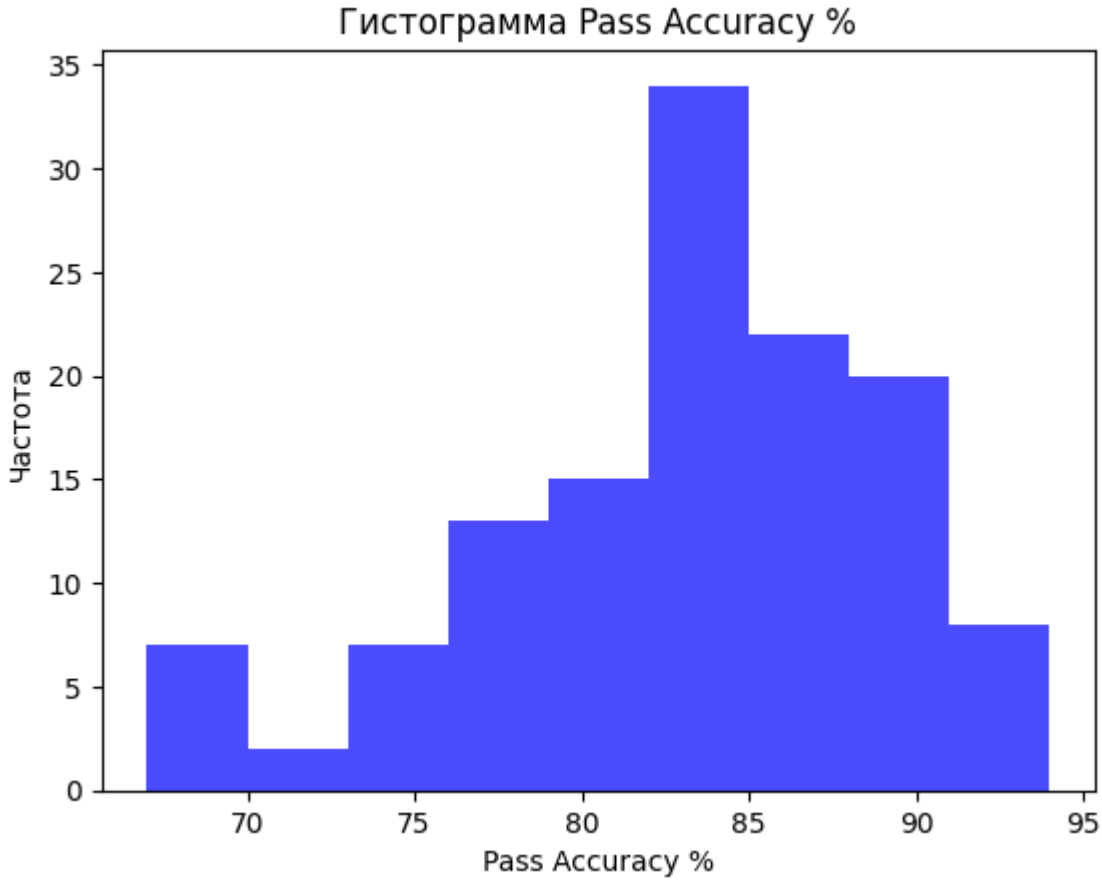

Гистограмма Blocked
Частота
Blocked
0
5
10
15
20
25
0
2
4
6
8
10

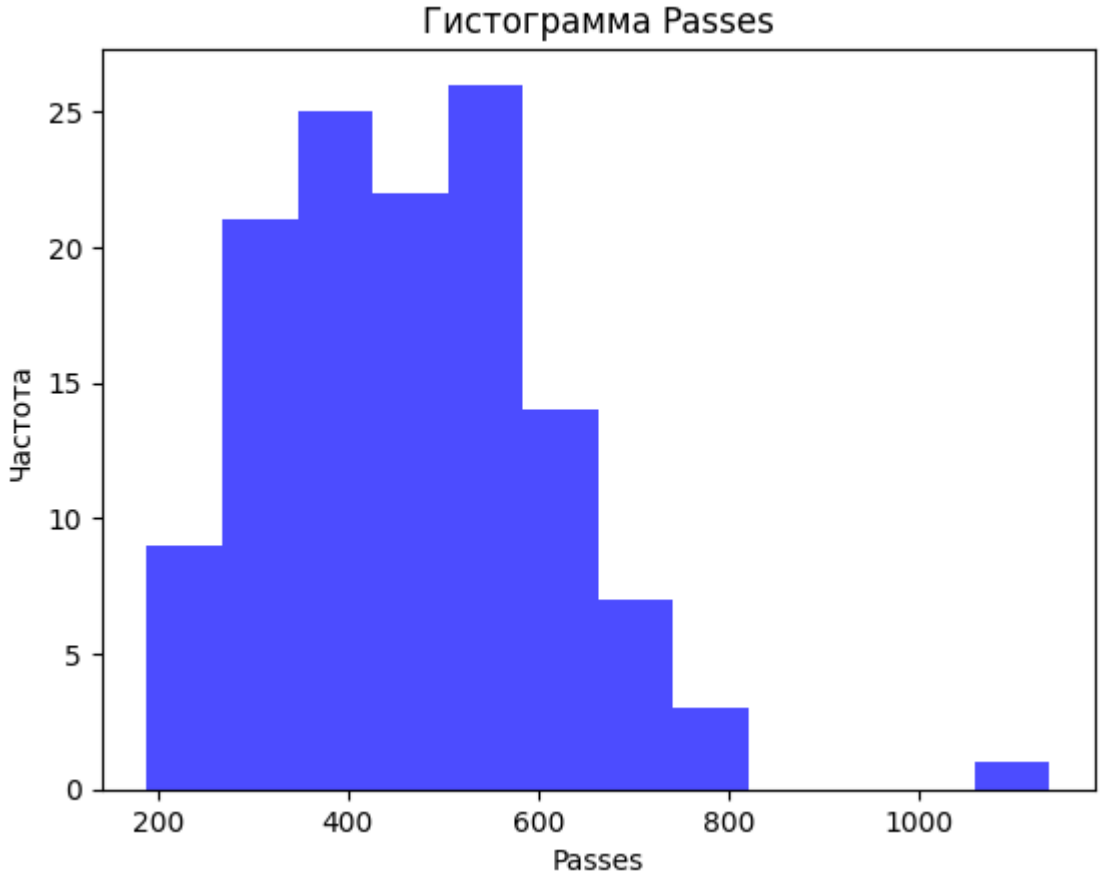

Гистограмма Corners
Частота
Corners
0
5
10
15
20
25
0
2
4
6
8
10

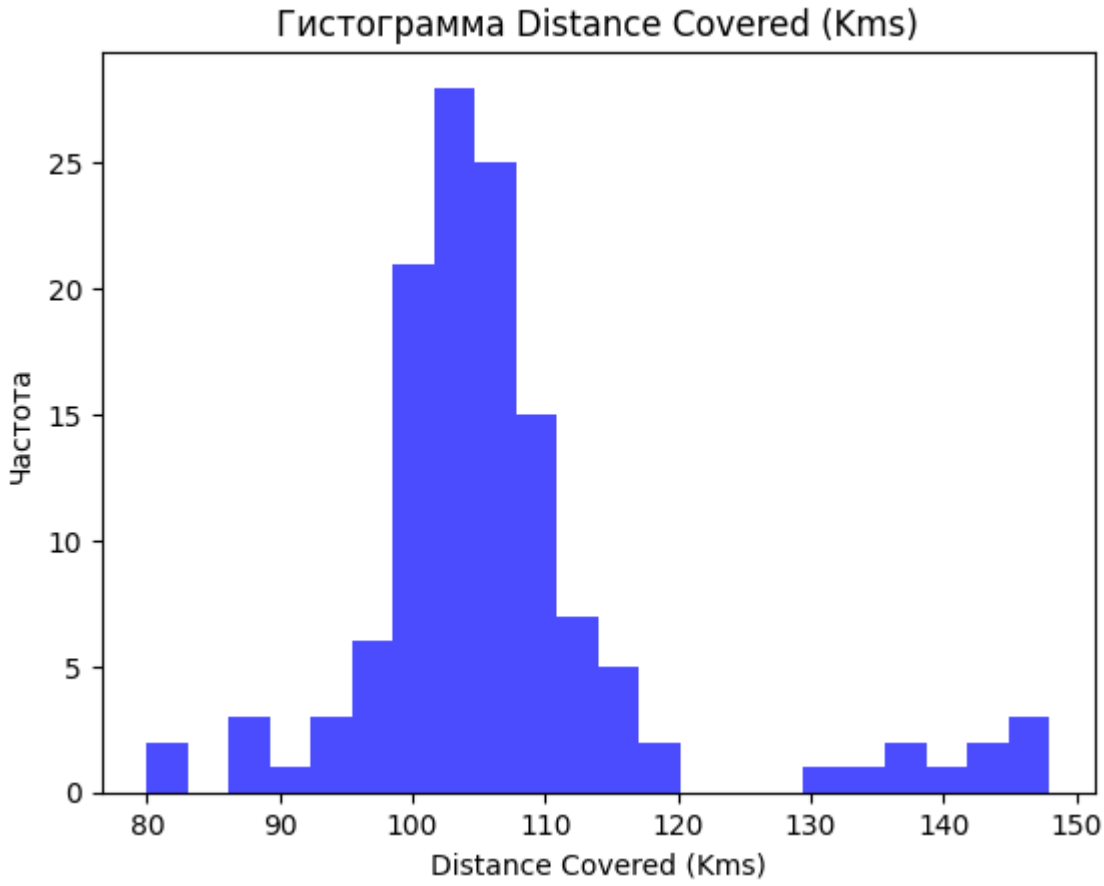

Гистограмма Offsides
Частота
Offsides
0
10
20
30
40
50
0
1
2
3
4
5

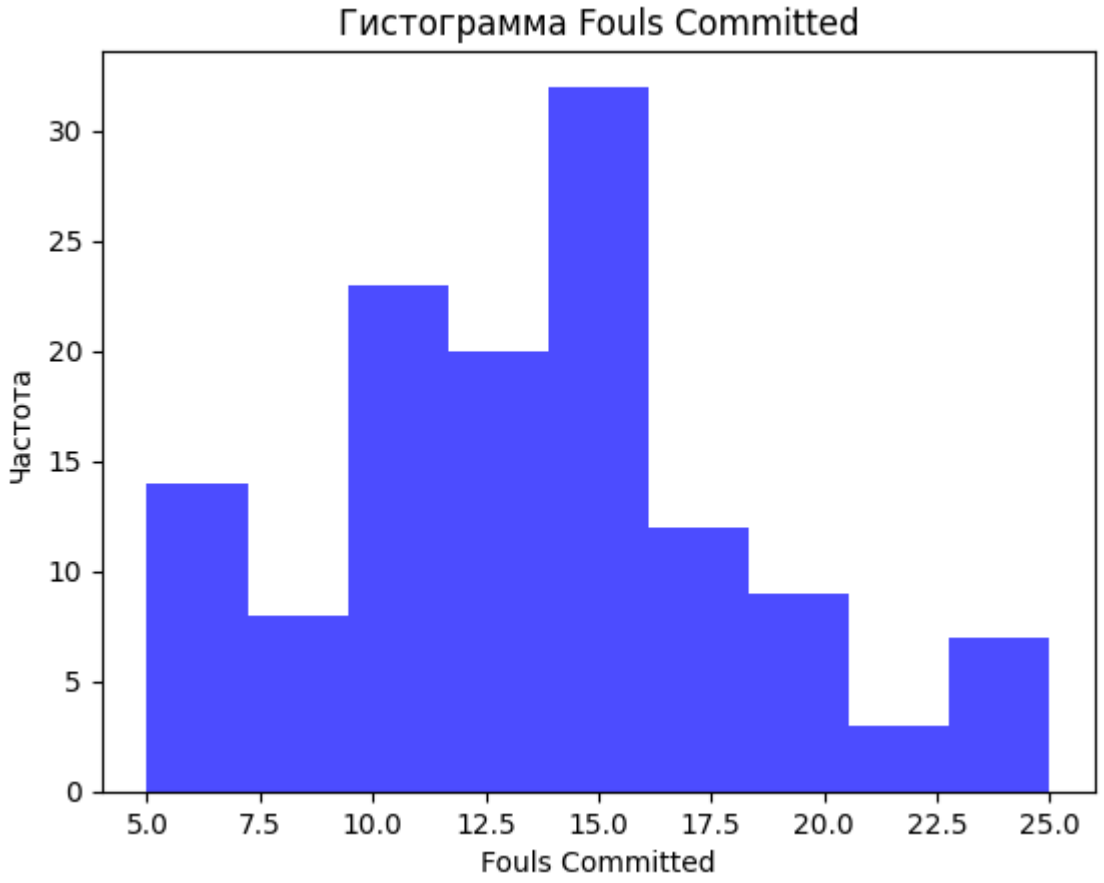

Гистограмма Free Kicks
Частота
Free Kicks
0
5
10
15
20
25
30
5
10
15
20
25

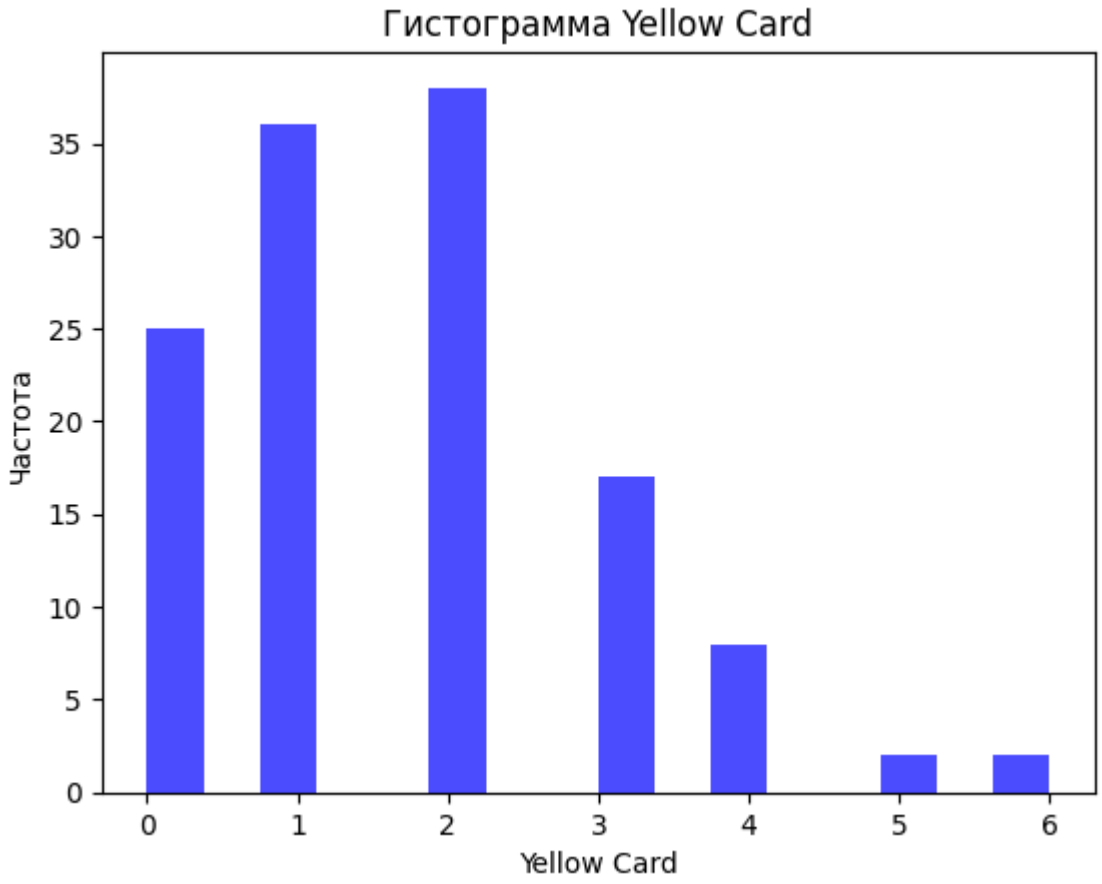

Гистограмма Saves
Частота
Saves
0
5
10
15
20
25
30
35
40
0
2
4
6
8

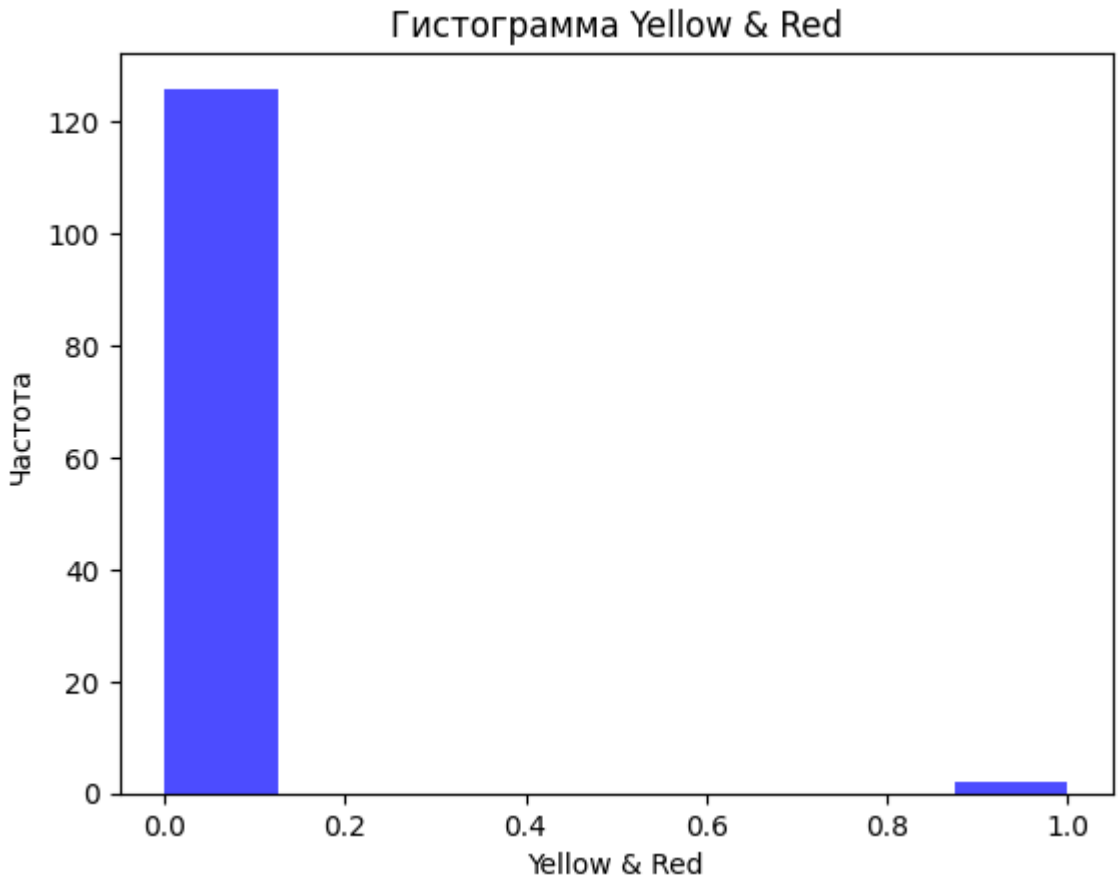

Гистограмма Pass Accuracy %
Частота
Pass Accuracy %
0
5
10
15
20
25
30
35
70
75
80
85
90
95

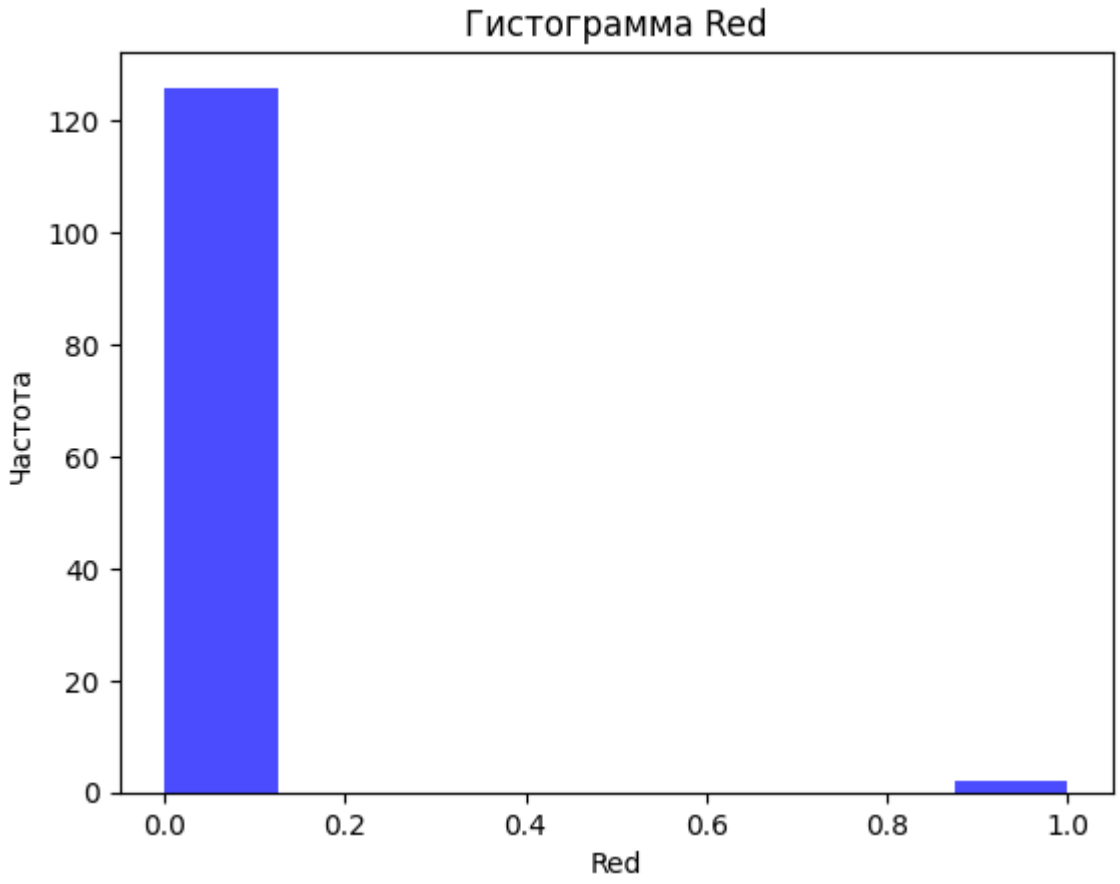

Гистограмма Passes
Частота
Passes
0
5
10
15
20
25
200
400
600
800
1000

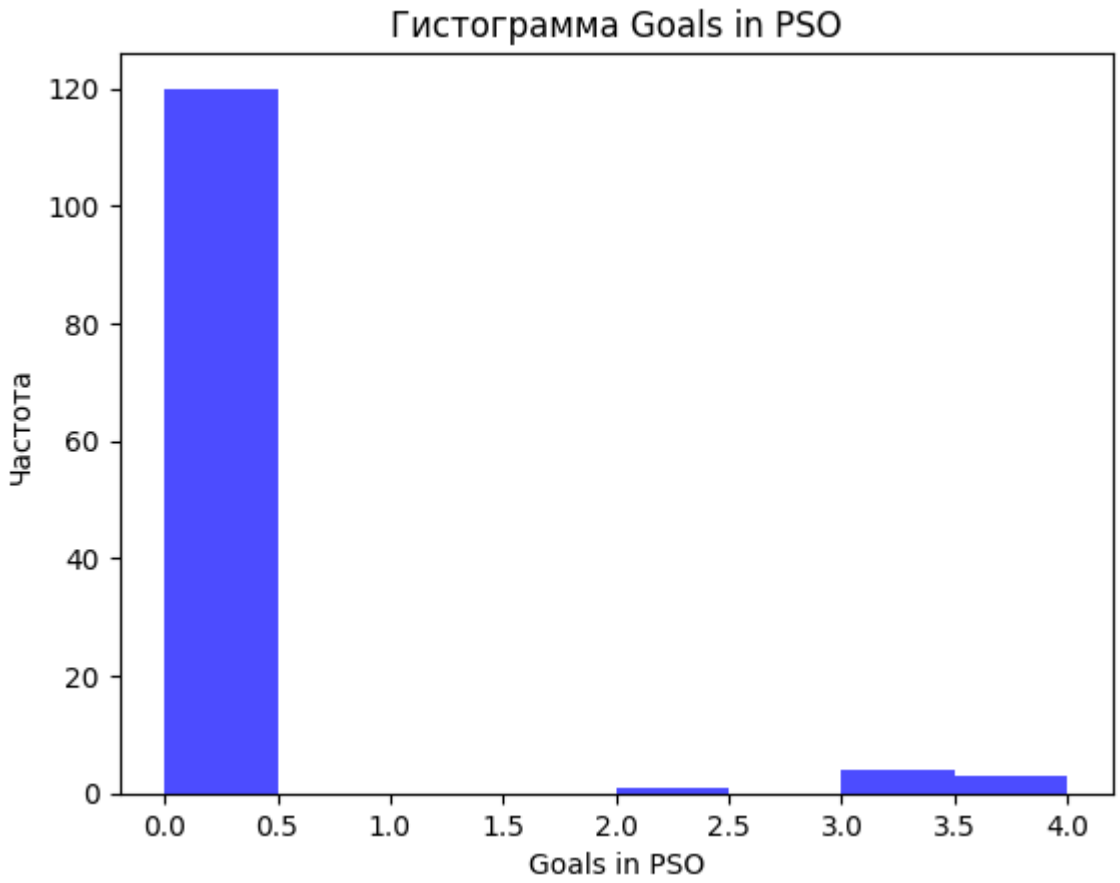

Гистограмма Distance Covered (Kms)
Частота
0
5
10
15
20
25
80
90
100
110
120
130
140
150
Distance Covered (Kms)


Гистограмма Fouls Committed
Частота
0
5
10
15
20
25
30
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
Fouls Committed


Гистограмма Yellow Card
Частота
0
5
10
15
20
25
30
35
0
1
2
3
4
5
6
Yellow Card


Гистограмма Yellow & Red
Частота
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0
Yellow & Red

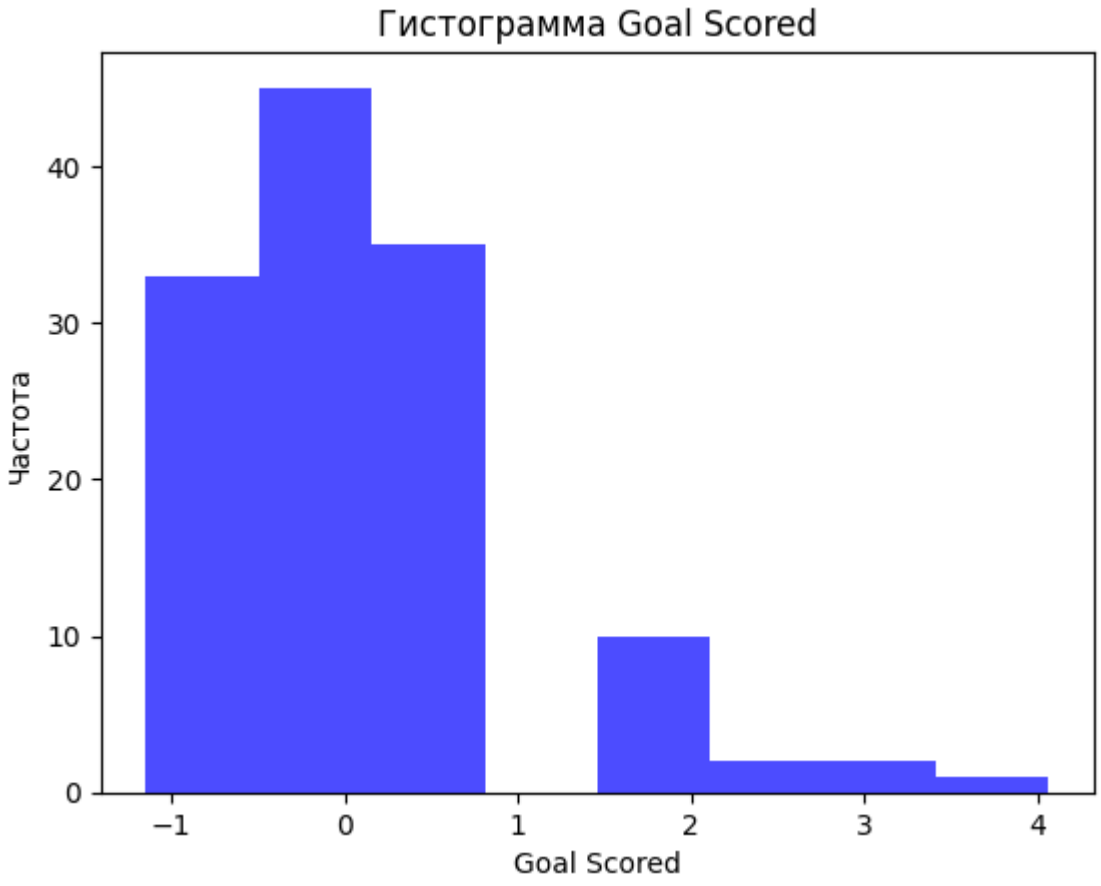


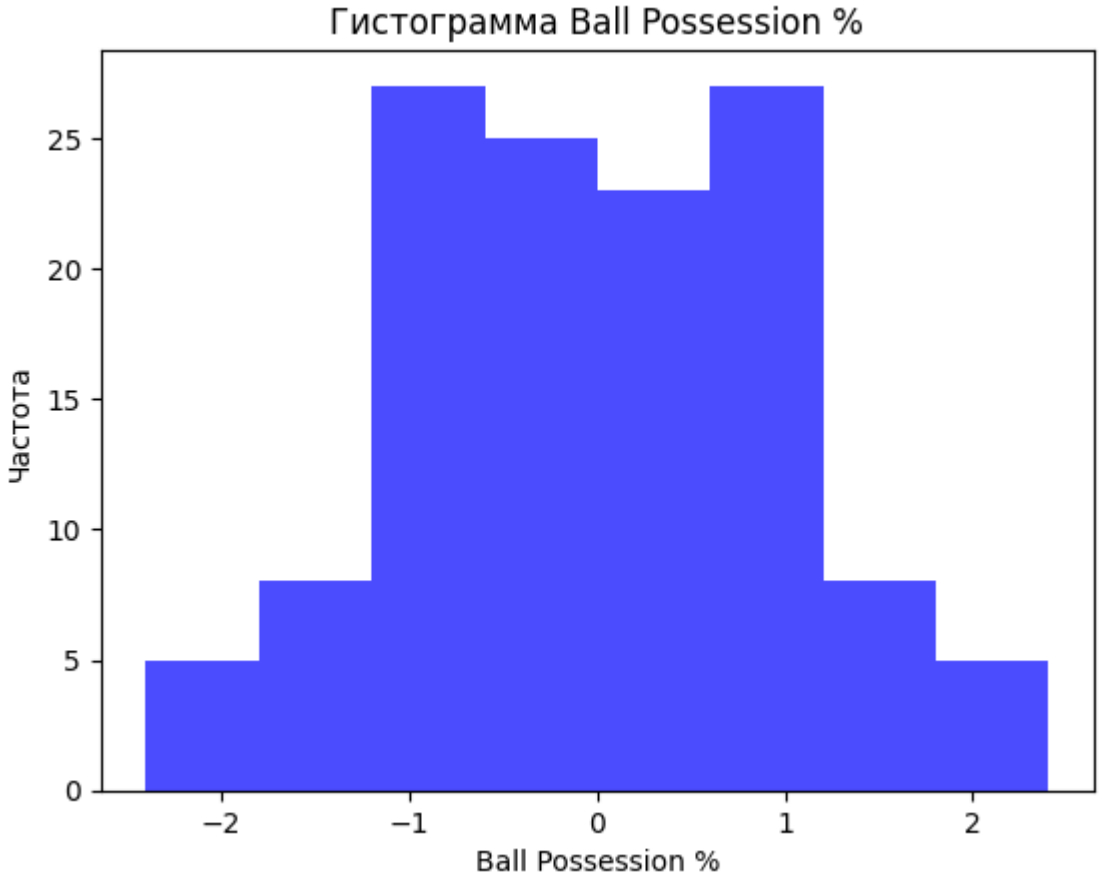


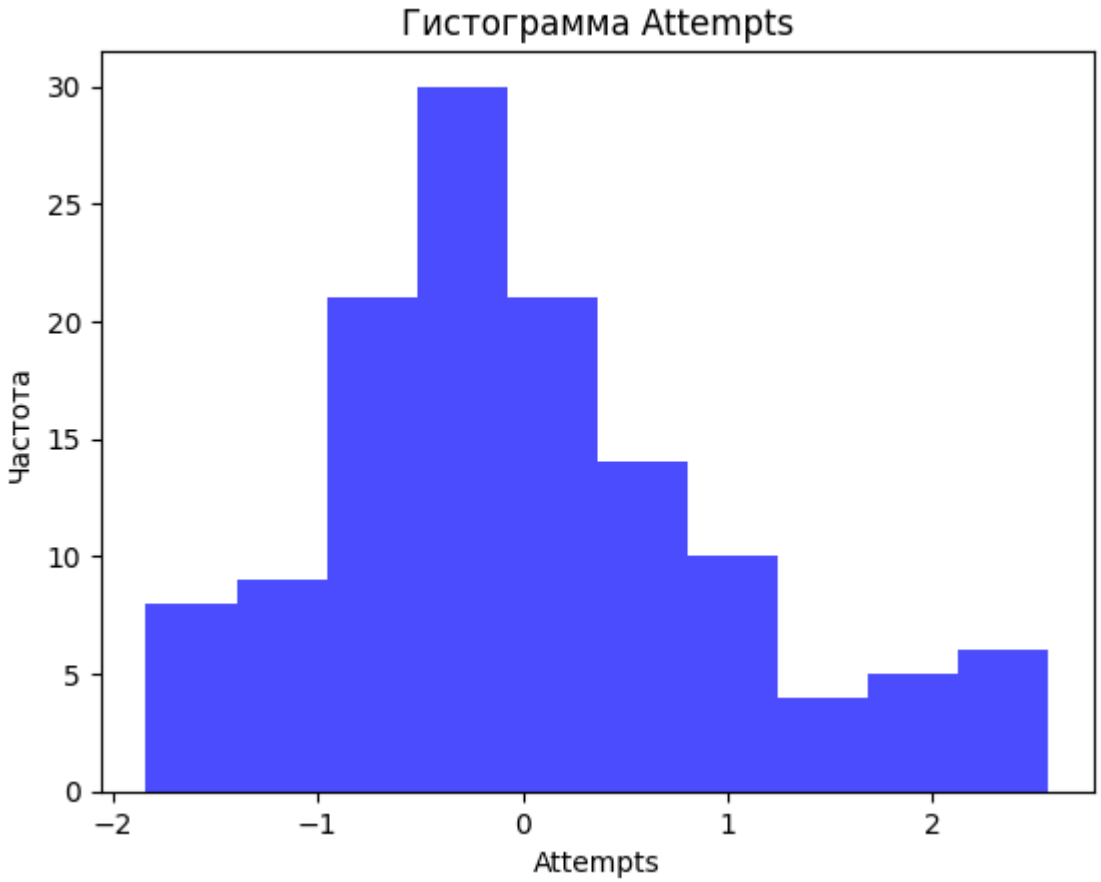
In [16]:
```
scaler = StandardScaler()
```

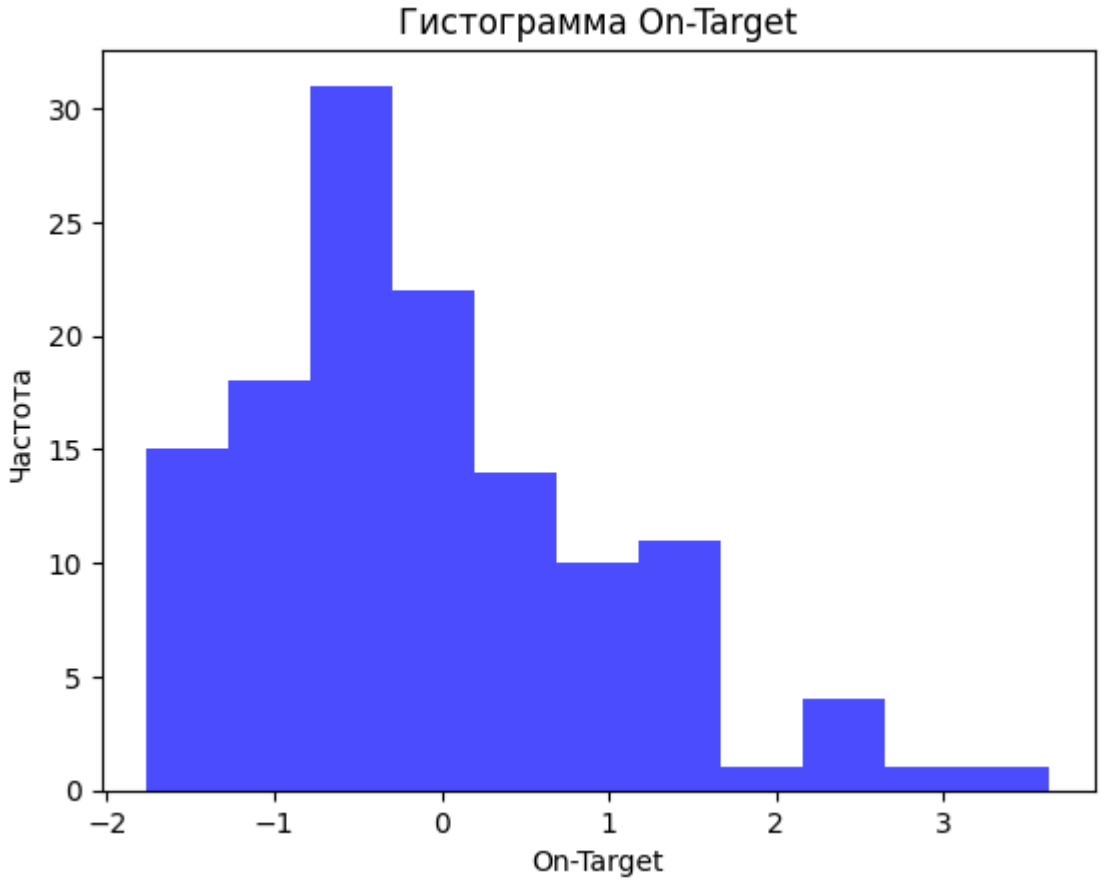
```
df_encoded[numerical_columns] = scaler.fit_transform(df_encoded[numerical_columns])
```

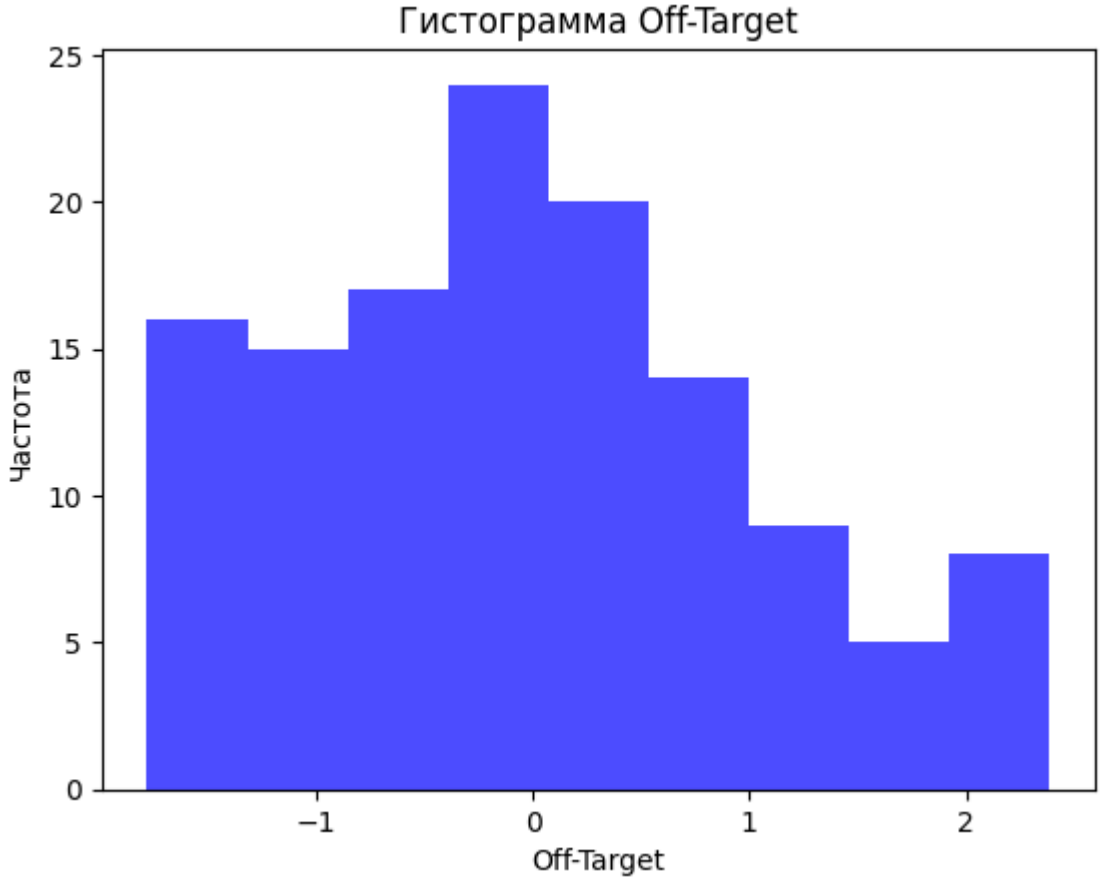
In [17]:
```
for column in numerical_columns:
    plt.figure()
    plt.hist(df_encoded[column], bins='auto', color='blue', alpha=0.7)
    plt.xlabel(column)
    plt.ylabel('Частота')
    plt.title(f'Гистограмма {column}')
    plt.show()
```

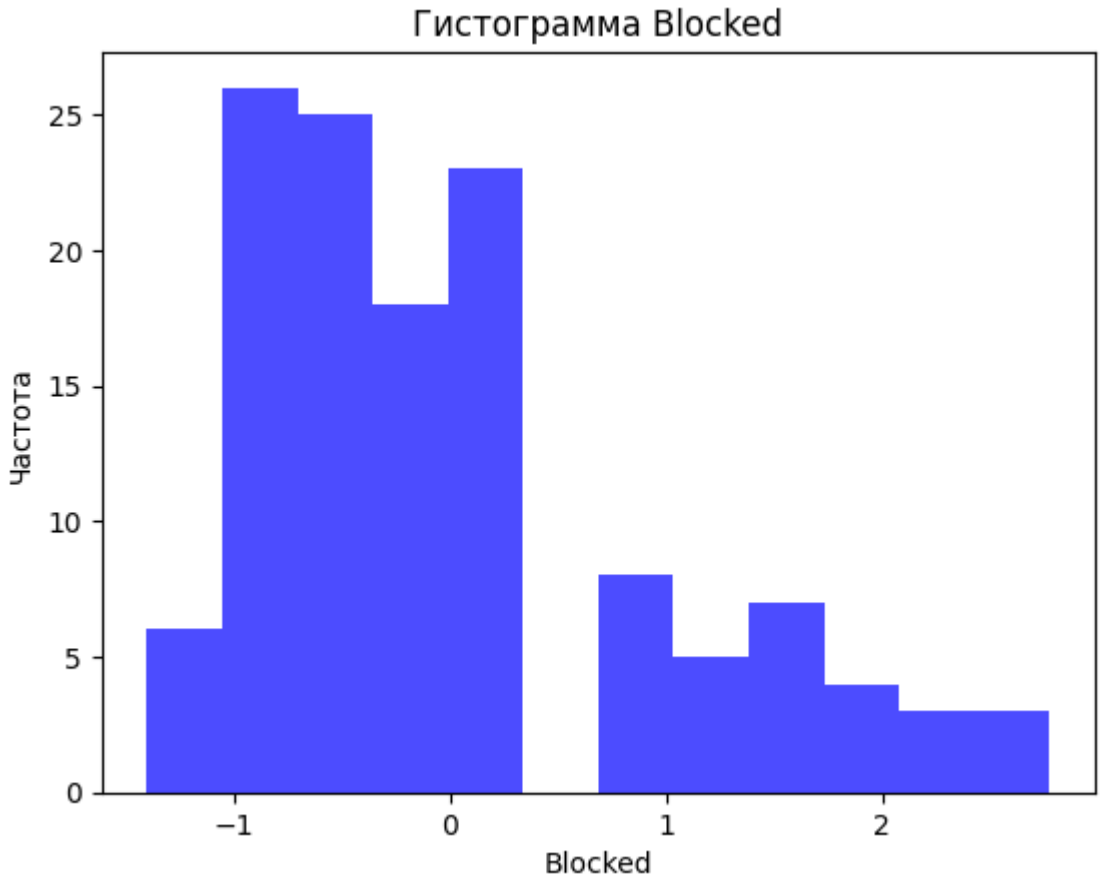


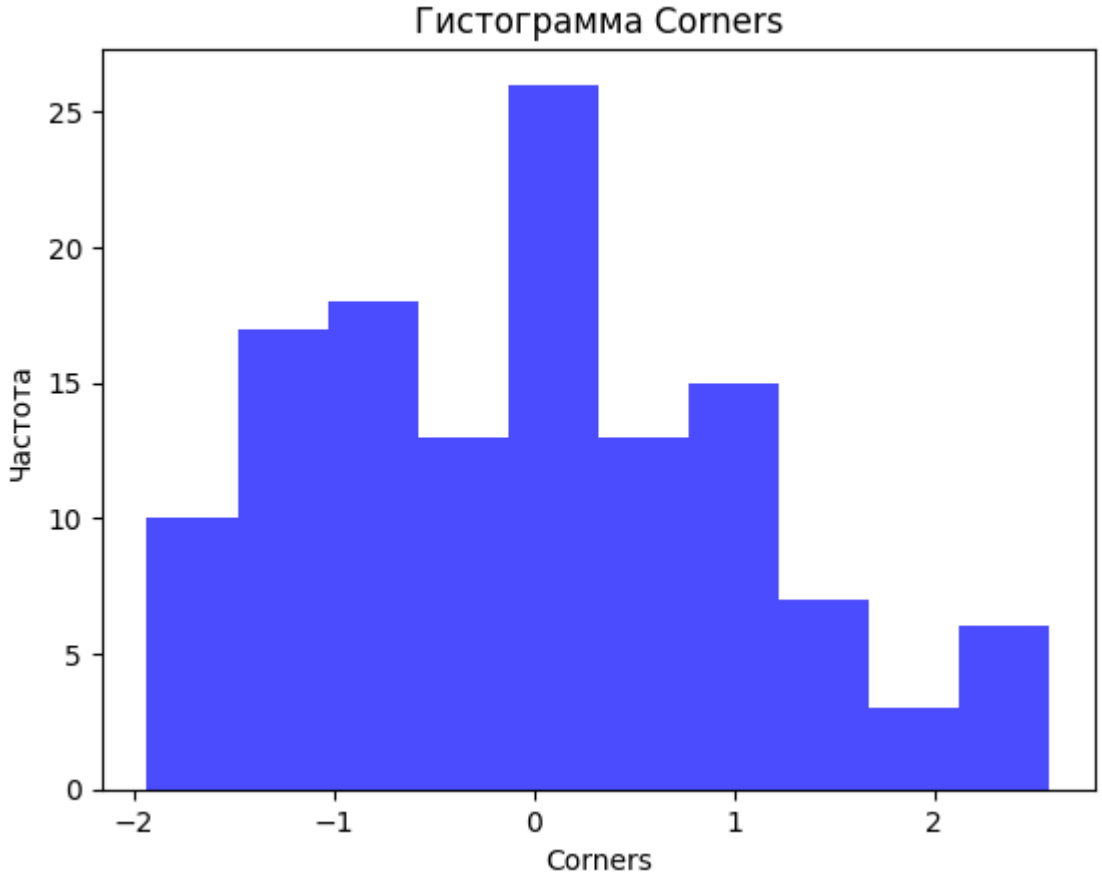

Гистограмма Ball Possession %
Частота
0
5
10
15
20
25
−2
−1
0
1
2
Ball Possession %

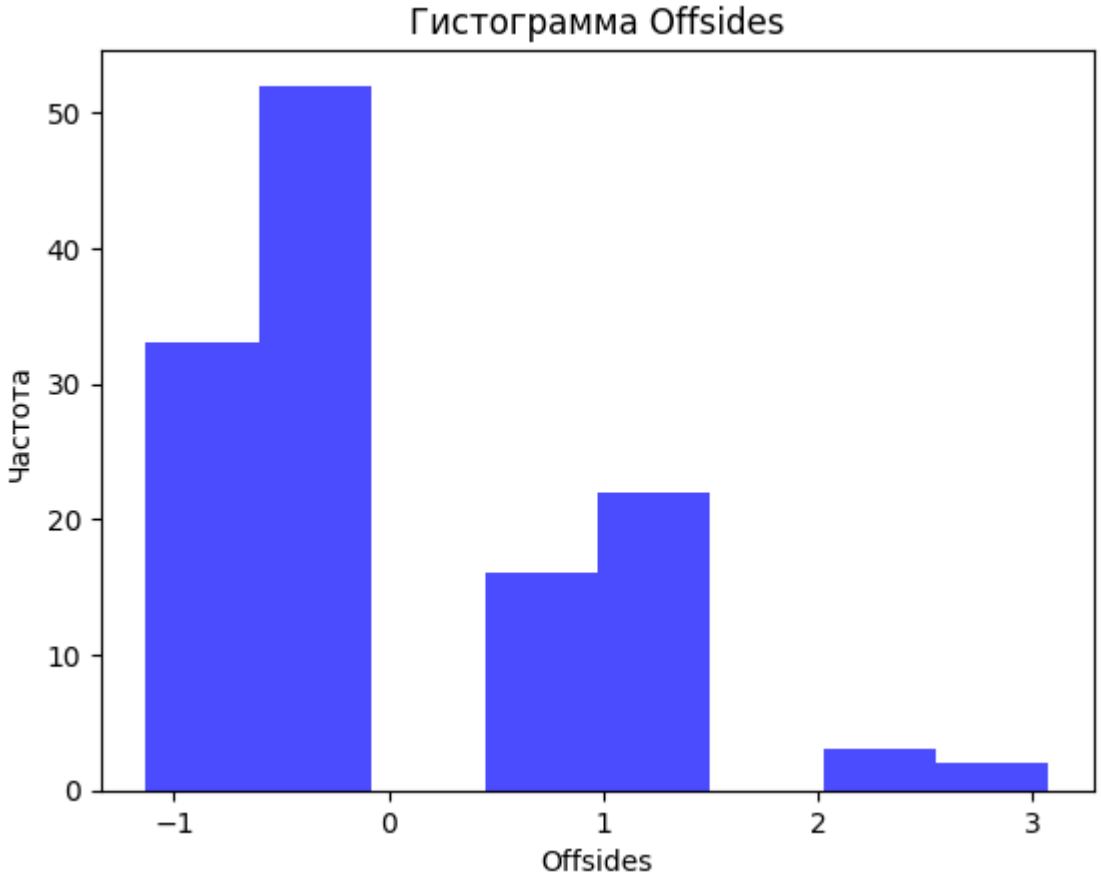

Гистограмма Attempts
Частота
0
5
10
15
20
25
30
−2
−1
0
1
2
Attempts

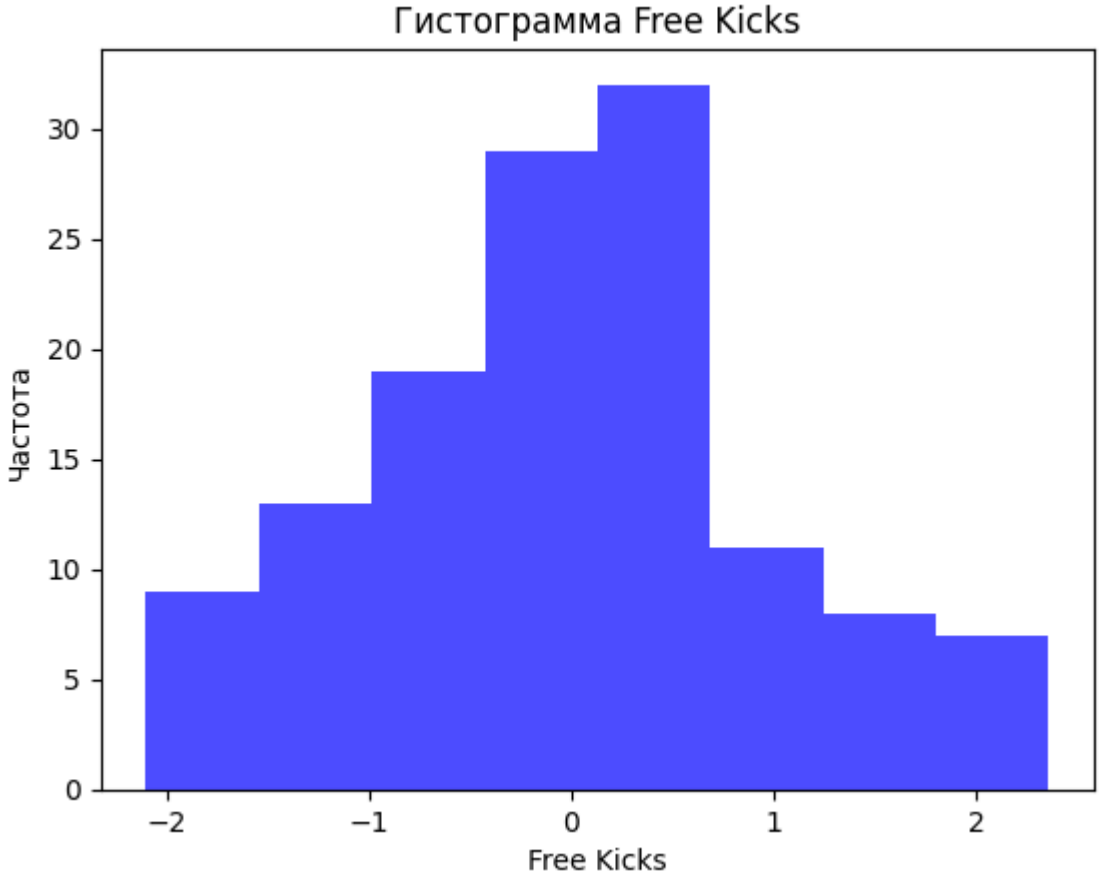

Гистограмма On-Target
Частота
On-Target
0
5
10
15
20
25
30
−2
−1
0
1
2
3

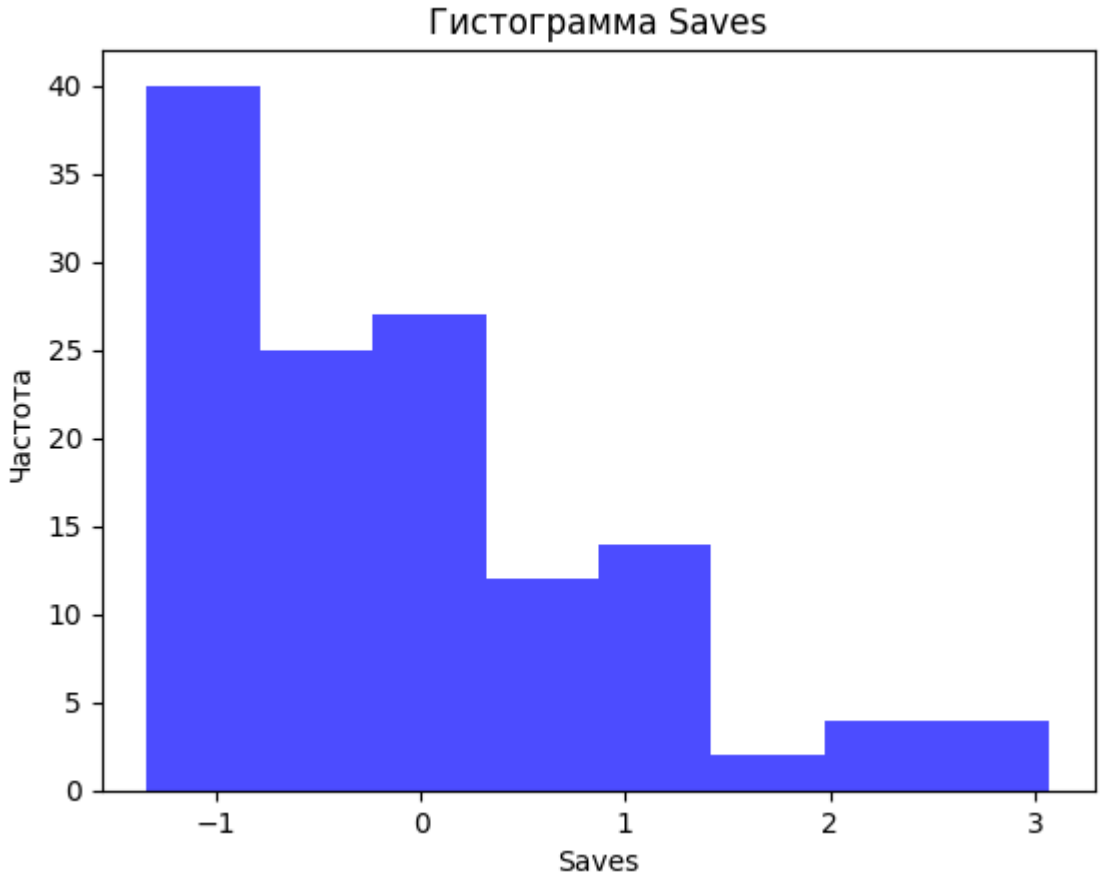

Гистограмма Off-Target
Частота
Off-Target
0
5
10
15
20
25
−1
0
1
2

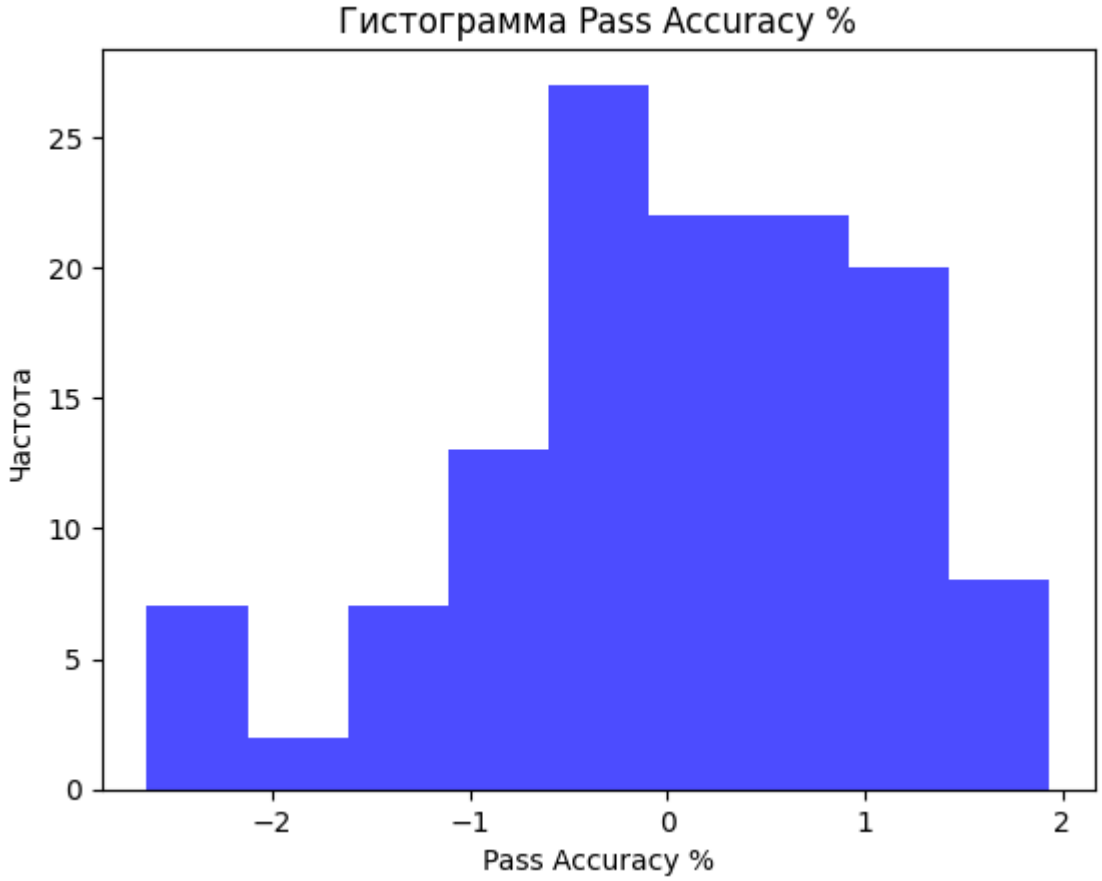

Гистограмма Blocked
Частота
Blocked
0
5
10
15
20
25
−1
0
1
2

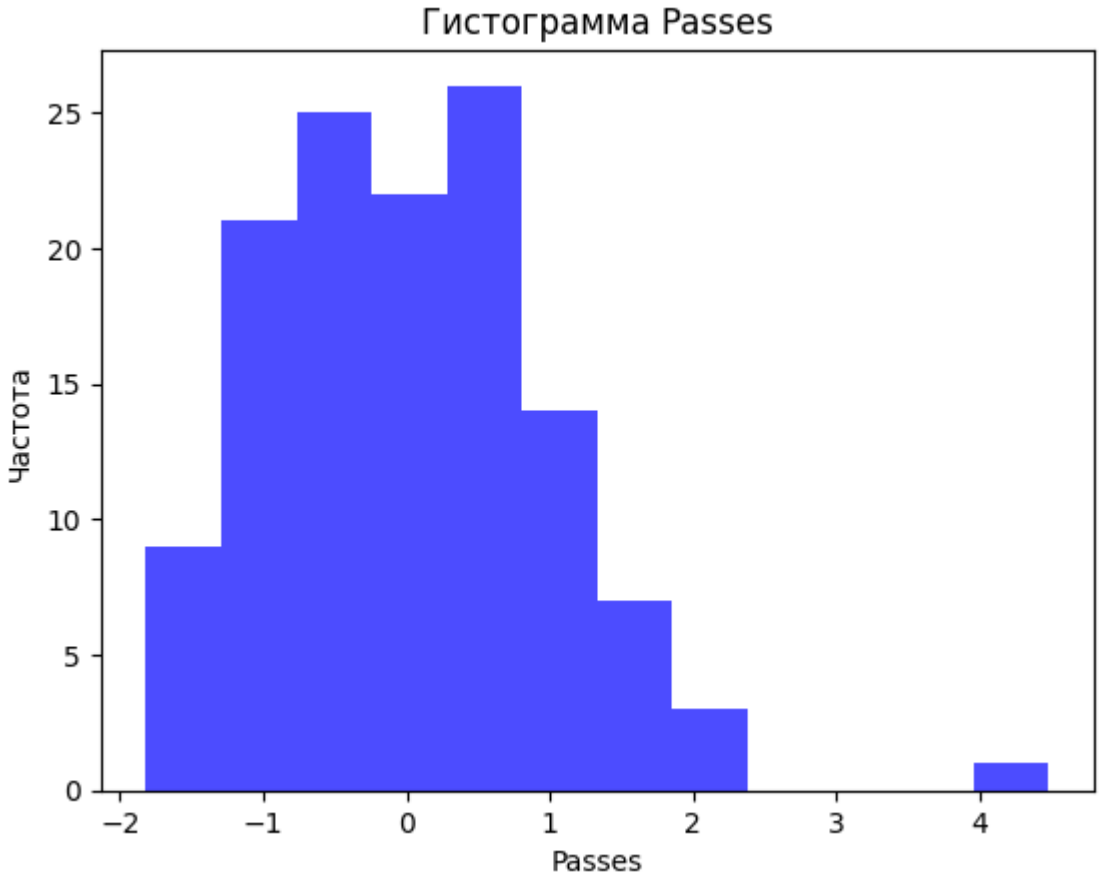

Гистограмма Corners
Частота
Corners
0
5
10
15
20
25
−2
−1
0
1
2

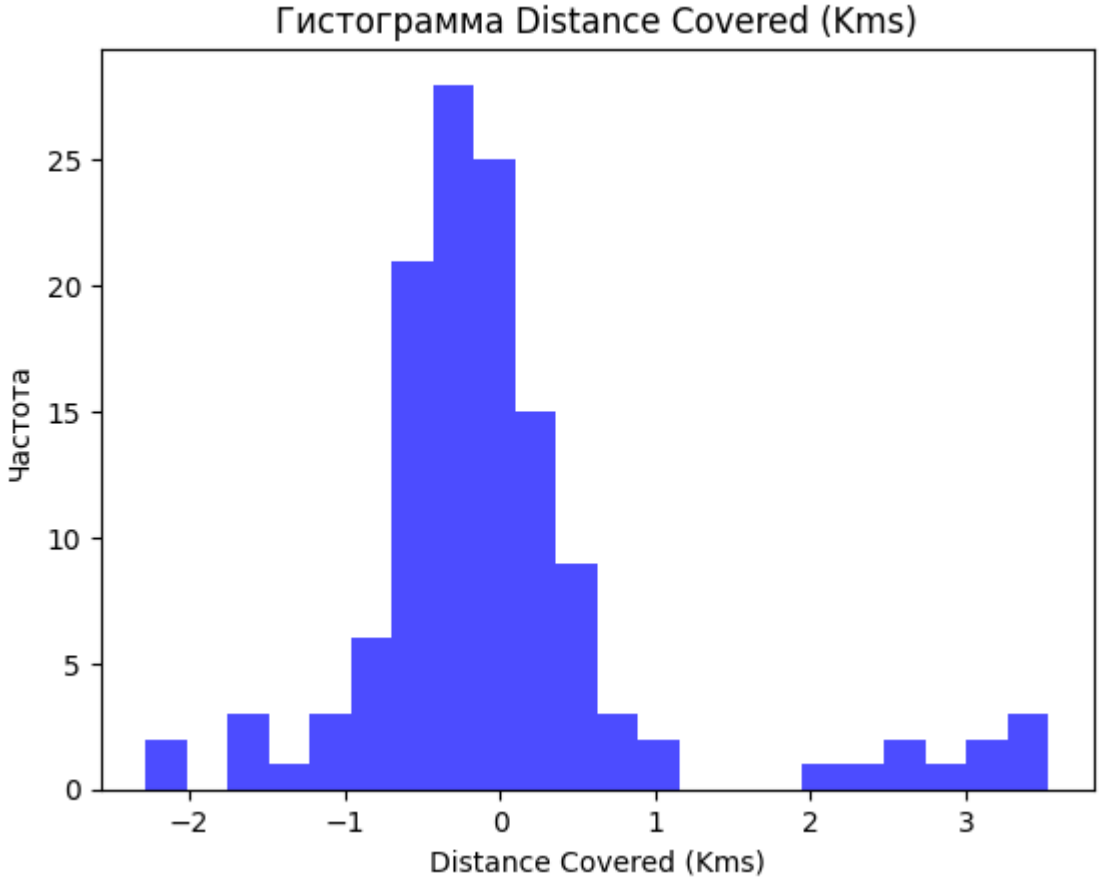

Гистограмма Offsides
Частота
50
40
30
20
10
0
−1
0
1
2
3
Offsides

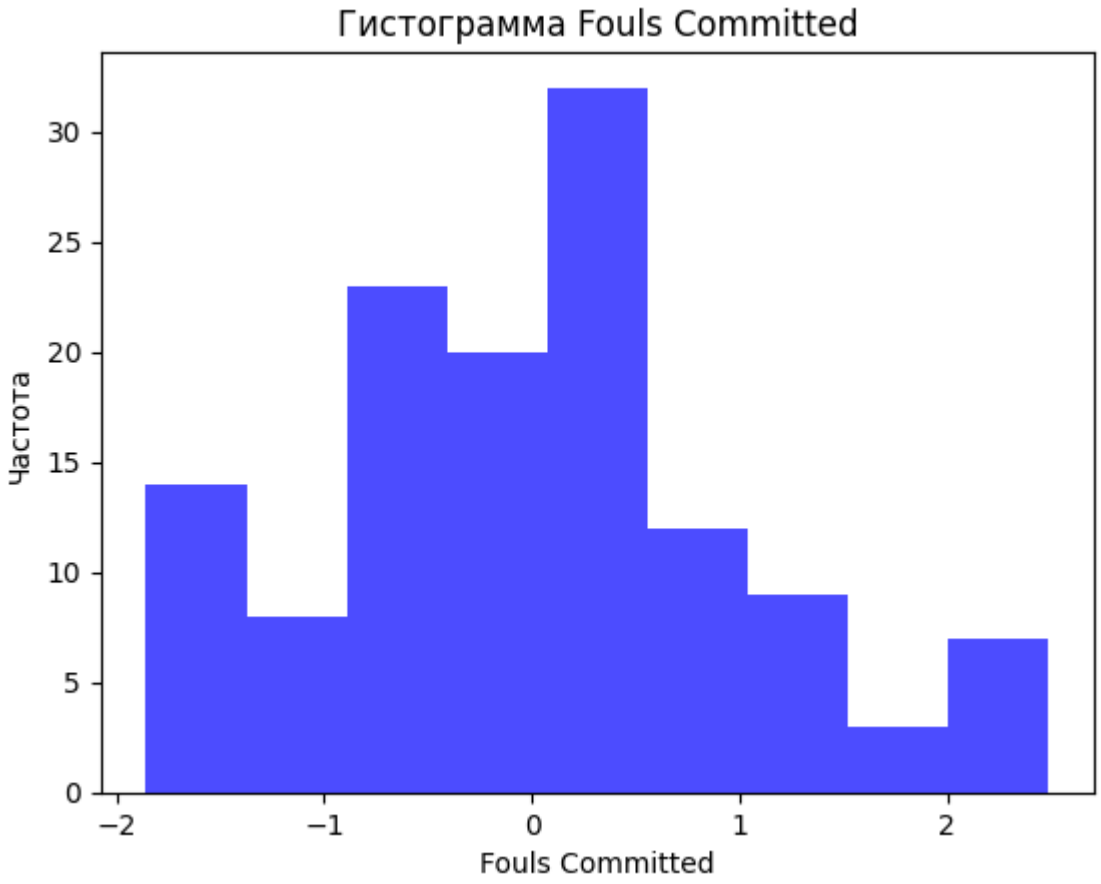

Гистограмма Free Kicks
Частота
30
25
20
15
10
5
0
−2
−1
0
1
2
Free Kicks

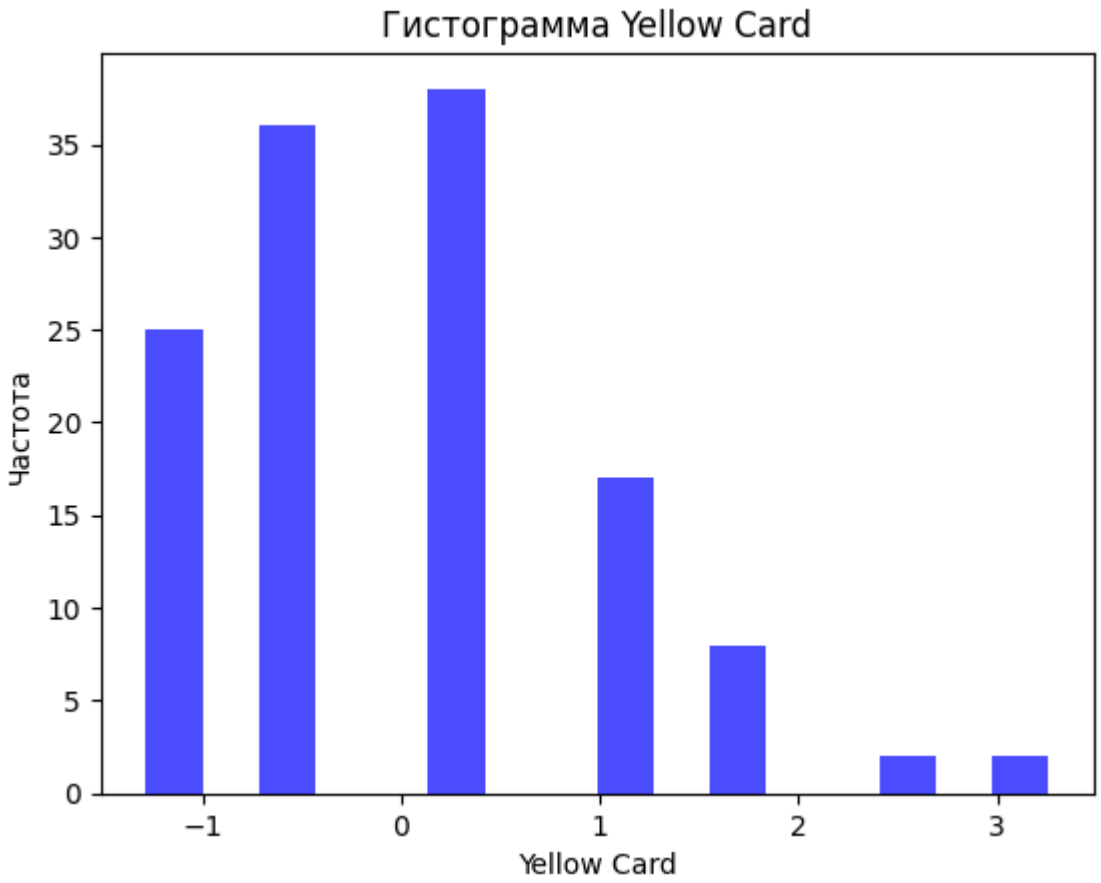

Гистограмма Saves
Частота
Saves
0
5
10
15
20
25
30
35
40
−1
0
1
2
3

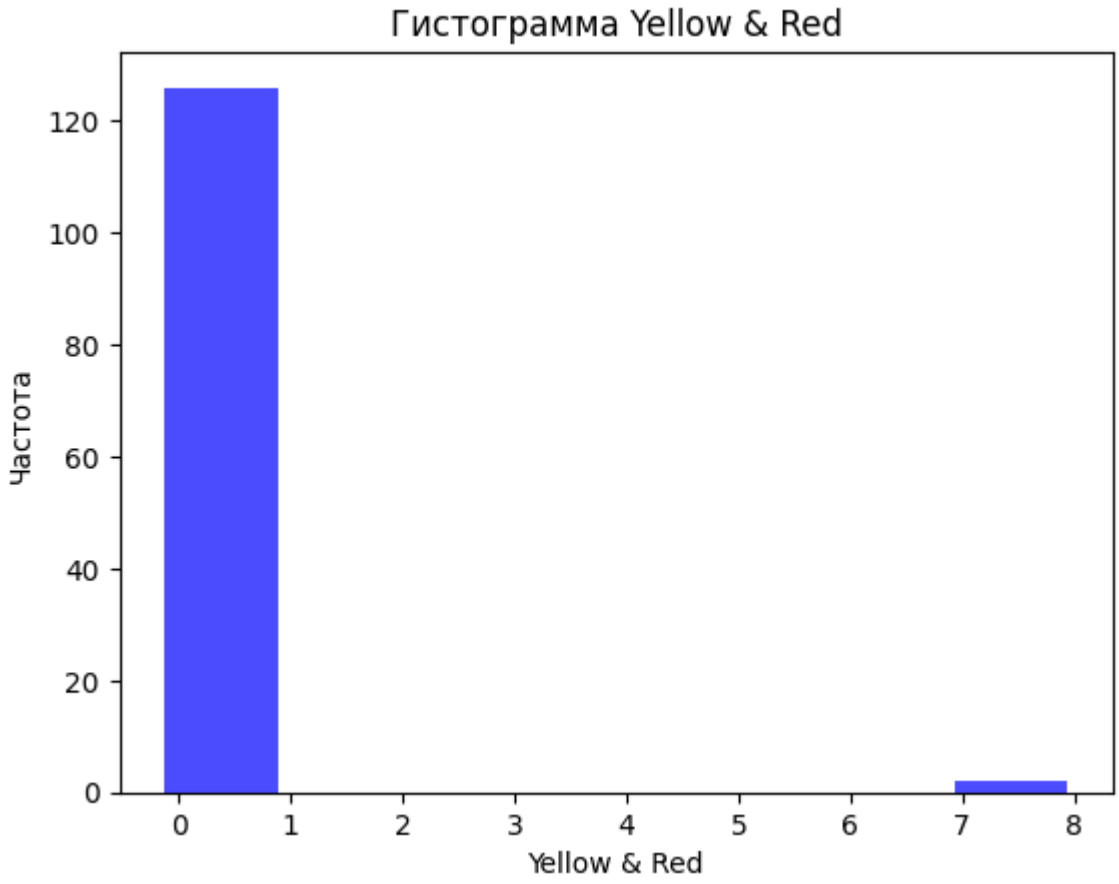

Гистограмма Pass Accuracy %
Частота
Pass Accuracy %
0
5
10
15
20
25
−2
−1
0
1
2

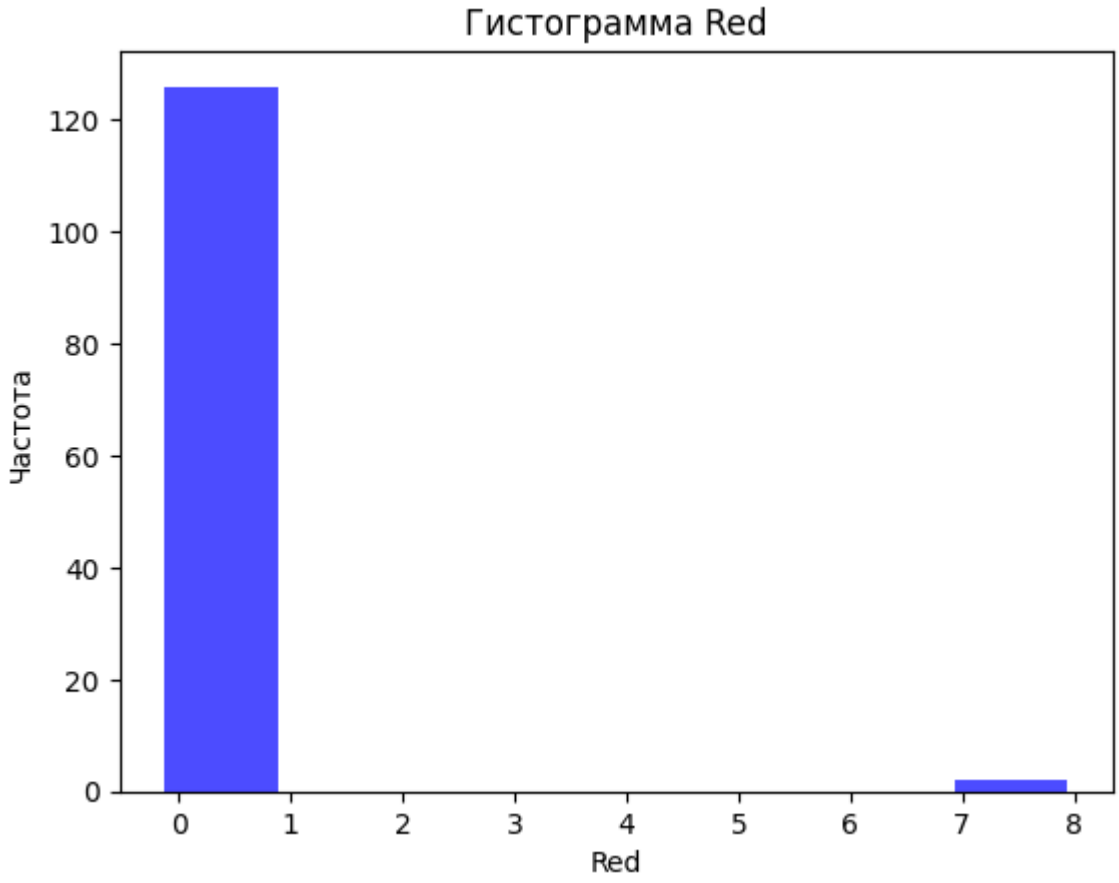

Гистограмма Passes
Частота
Passes
0
5
10
15
20
25
−2
−1
0
1
2
3
4

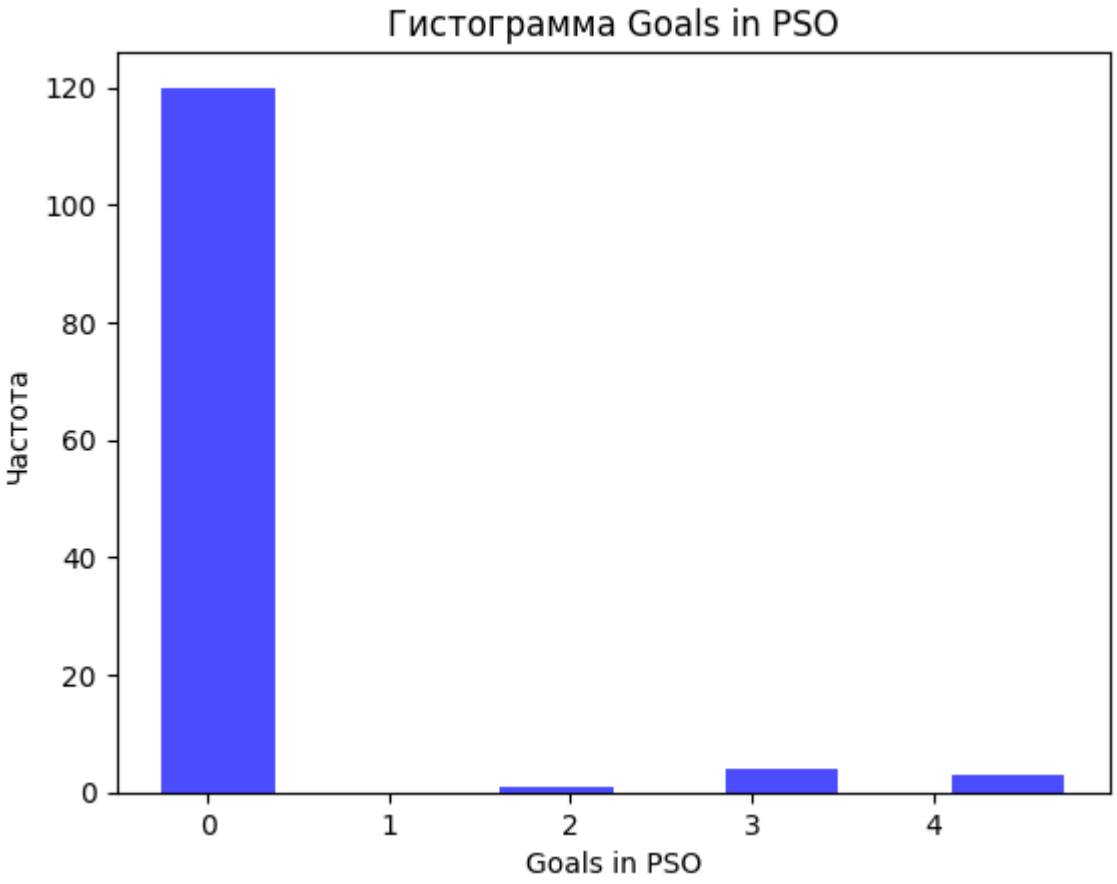

Гистограмма Distance Covered (Kms)
Частота
Distance Covered (Kms)
0
5
10
15
20
25
−2
−1
0
1
2
3


Гистограмма Fouls Committed
Частота
Fouls Committed
0
5
10
15
20
25
30
−2
−1
0
1
2


Гистограмма Yellow Card
Частота
Yellow Card
0
5
10
15
20
25
30
35
−1
0
1
2
3


Гистограмма Yellow & Red
Частота
0
20
40
60
80
100
120
0
1
2
3
4
5
6
7
8
Yellow & Red


Гистограмма Red
Частота
0
20
40
60
80
100
120
0
1
2
3
4
5
6
7
8
Red



Признаки отмасштабированы, распределение не изменилось

Выбор метрик

1. ROC-AUC:

$ROCAUC = \int_0^1 \mathrm{TP}(t)\,\mathrm{dFP}(t)$, где TP - доля истинно положительных (True Positive Rate), FP - доля ложно положительных (False Positive Rate), и интеграл берется по кривой ROC. ROC-AUC является мерой качества модели, основанной на ее способности разделять классы.

2. Recall:

$Recall = \frac{TP}{TP+FN}$, где TP - число истинно положительных, FN - число ложно отрицательных. Измеряет долю верно предсказанных положительных значений относительно всех реальных положительных значений. Полнота полезна, когда важно минимизировать ложно отрицательные предсказания.

3. Specificity:

$Specificity = \frac{TN}{TN+FP}$ где TN - число истинно отрицательных, FP - число ложно положительных. Измеряет долю верно предсказанных отрицательных значений

относительно всех реальных отрицательных значений. Полезна в данном случае, т.к. присуствует дисбаланс классов, т.е. значений "0" меньше чем "1".

Обучение моделей

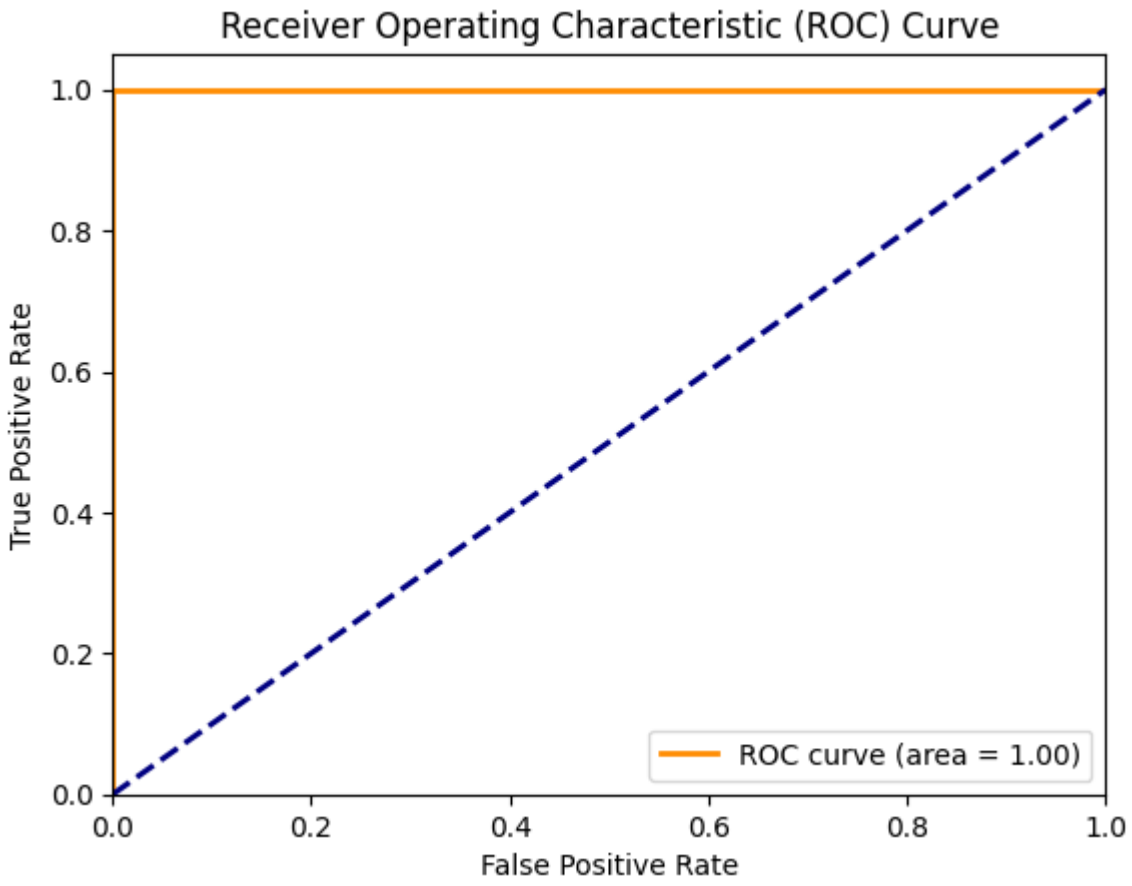
In [18]:
```
def results(X_test, y_test, model):
  y_pred_proba = model.predict_proba(X_test)[:, 1]
  y_pred = model.predict(X_test)

  # Вычисление ROC-кривой
  fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
  roc_auc = auc(fpr, tpr)

  # Recall
  recall = recall_score(y_test, y_pred)

  # Построение ROC-кривой
  plt.figure()
  plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % r
  plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver Operating Characteristic (ROC) Curve')
  plt.legend(loc="lower right")
  plt.show()

  tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
  specificity = tn / (tn+fp)

  # Вывод результатов
  print("ROC-AUC:", roc_auc)
  print("Recall:", recall)
  print("Specificity:", specificity)
```

Разделение на обучающую и тестовую выборку

In [19]:
```
X = df_encoded.drop(columns=['Man of the Match_Yes'])
y = df_encoded['Man of the Match_Yes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_st
```

Логистическая регрессия

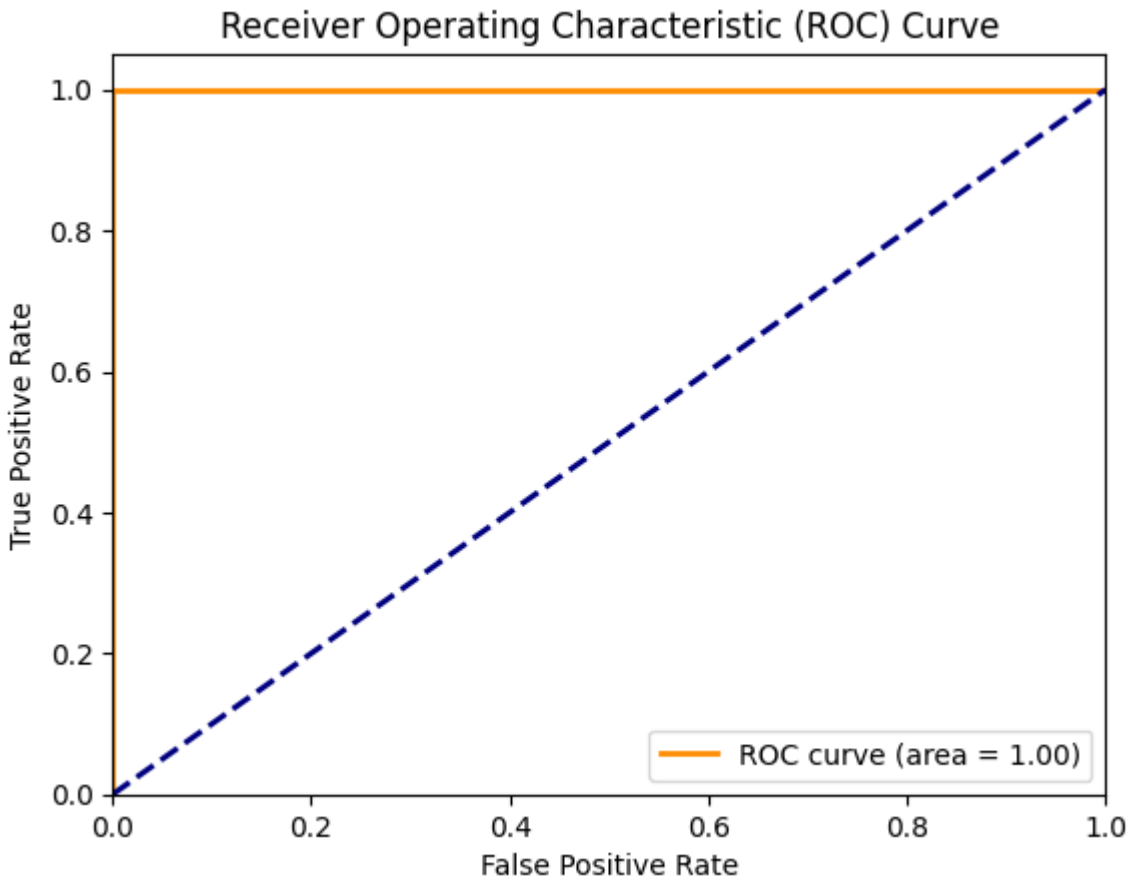
In [20]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
results(X_test, y_test, model)
```




```
ROC-AUC: 1.0
Recall: 1.0
Specificity: 1.0
```

Случайный лес

In [21]:
```
model = RandomForestClassifier()
model.fit(X_train, y_train)
results(X_test, y_test, model)
```




```
ROC-AUC: 1.0
Recall: 1.0
Specificity: 1.0
```

Вывод

Проведенное обучение модели с метриками ROC-AUC = 1, Recall = 1 и Specificity = 1 говорит о высоком качестве модели и ее способности точно классифицировать данные.

Значение ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) равное 1 означает, что модель идеально разделяет положительные и отрицательные примеры, без ложных срабатываний или пропусков.

Recall, также известный как чувствительность, равный 1 указывает на то, что модель правильно классифицирует все положительные примеры, не допуская ложноотрицательных результатов.

Specificity, равный 1, означает, что модель правильно классифицирует все отрицательные примеры, не допуская ложноположительных результатов.

Таким образом, при данных значениях метрик можно сделать вывод, что модель обладает идеальной способностью разделять классы и не допускает ни

ложноположительных, ни ложноотрицательных результатов. Это свидетельствует о высокой точности и надежности модели.