РК1 ММО

Студент: Сафин Рустам Равильевич

Группа: ИУ5-21М

Вариант в группе: 12

Вариант задачи №1 (Задание 12): Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "логарифм - np.log(X)".

Вариант задачи №2 (Задание 32): Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), обратный алгоритм (sequential backward selection).

Дополнительное задание по группам: Для пары произвольных колонок данных построить график "Диаграмма рассеяния".

```
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
%matplotlib inline
sns.set(style="ticks")
```

Задача №1 (№12)

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "логарифм - np.log(X)".

```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

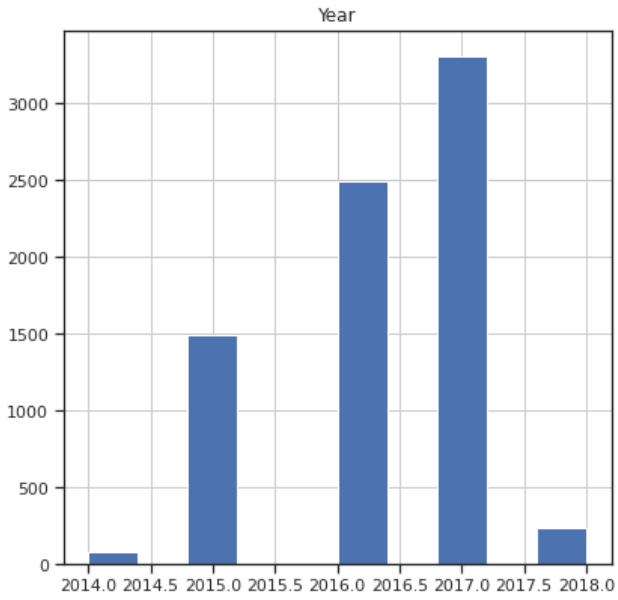
```
data = pd.read_csv("/content/data/LeagueofLegends.csv", sep=",")
```

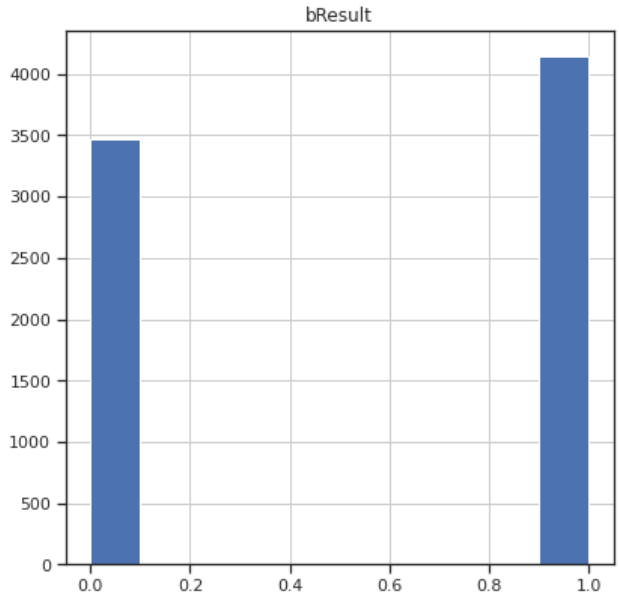
```
data.head()
```

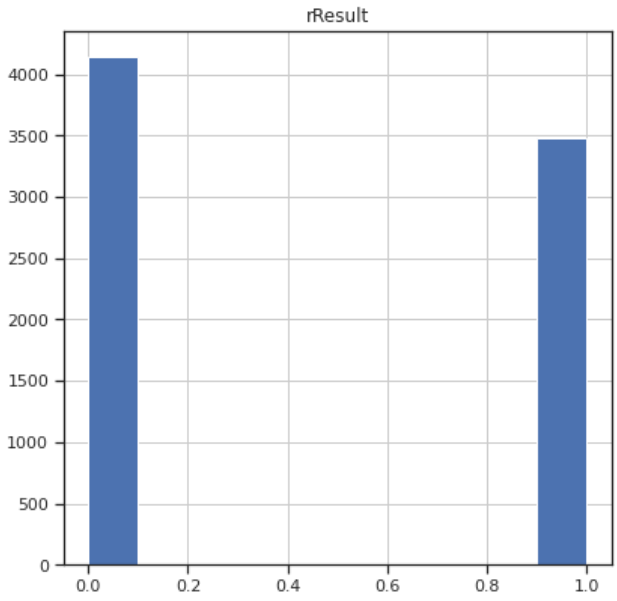
	League	Year	Season	Type	blueTeamTag	bResult	rResult	redTeamTag	gamelength
0	NALCS	2015	Spring	Season	TSM	1	0	C9	40
1	NALCS	2015	Spring	Season	CST	0	1	DIG	38
2	NALCS	2015	Spring	Season	WFX	1	0	GV	40
3	NALCS	2015	Spring	Season	TIP	0	1	TL	41
4	NALCS	2015	Spring	Season	CLG	1	0	T8	35

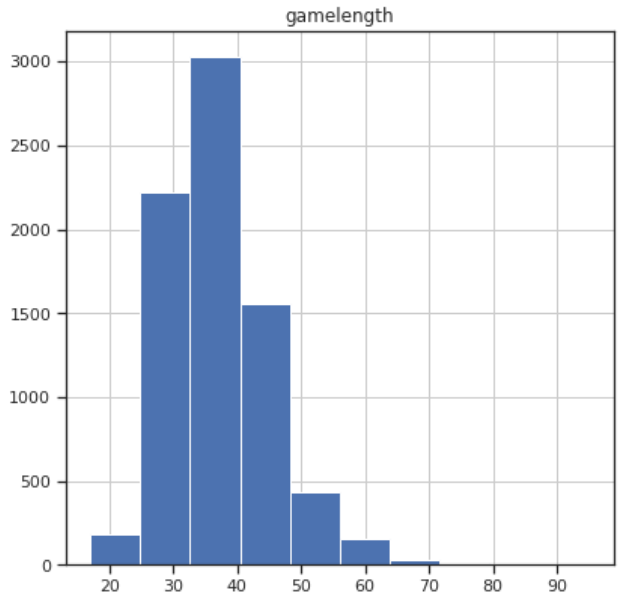
5 rows × 57 columns


```
data.hist(figsize=(15,15))
plt.show()
```



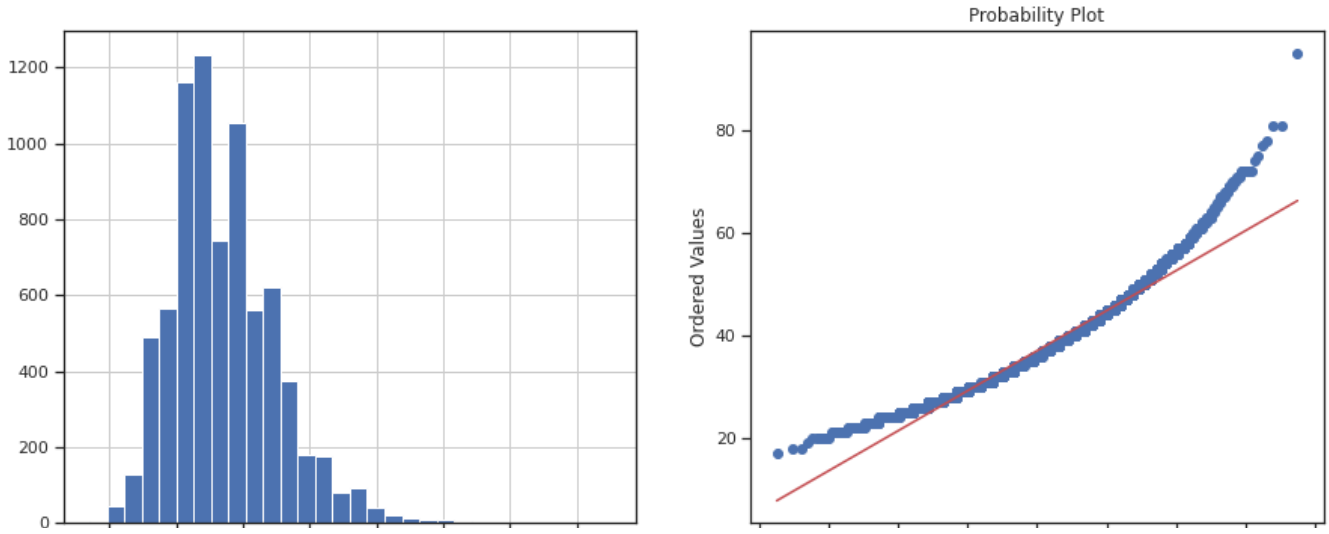






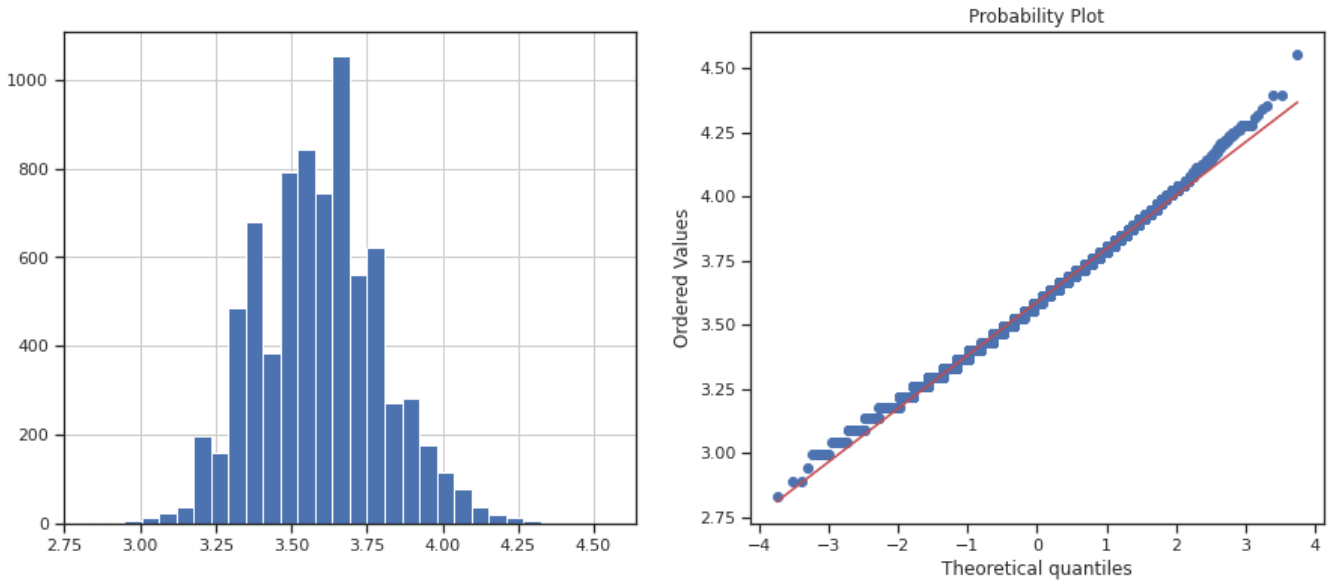
Посмотрим на исходное распределение одного из признаков.

```
diagnostic_plots(data, 'gamelength')
```



Как можно заметить, распределение этого признака немного отличается от нормального. Попробуем провести нормализацию признака с помощью функции логарифма.

```
data['gamelength_log'] = np.log(data['gamelength'])
diagnostic_plots(data, 'gamelength_log')
```



Таким образом, мы получили нормальное распределение этого числового признака.

Задача №2 (№32)

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), обратный алгоритм (sequential backward selection).

```
wine = load_wine()
wine_X = wine.data
wine_y = wine.target
wine_feature_names = wine['feature_names']
wine_x_df = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
```

```
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

knn = KNeighborsClassifier(n_neighbors=3)

sbs = SFS(knn,
          k_features=1,
          forward=False,
          floating=False,
          verbose=2,
          scoring='accuracy',
          cv=4)
sbs = sbs.fit(wine_X, wine_y)
```

```
sbs.subsets_
```

```
       'cv_scores': array([0.82222222, 0.95555556, 1.        , 0.97674419]),
       'feature_idx': (0, 3, 6, 8, 9),
       'feature_names': ('0', '3', '6', '8', '9')},
      6: {'avg_score': 0.9500000000000001,
       'cv_scores': array([0.86666667, 0.95555556, 0.97777778, 1.        ]),
       'feature_idx': (0, 3, 6, 8, 9, 11),
       'feature_names': ('0', '3', '6', '8', '9', '11')},
      7: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.86666667, 0.95555556, 1.        , 1.        ]),
       'feature_idx': (0, 2, 3, 6, 8, 9, 11),
       'feature_names': ('0', '2', '3', '6', '8', '9', '11')},
      8: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.86666667, 0.95555556, 1.        , 1.        ]),
       'feature_idx': (0, 2, 3, 6, 7, 8, 9, 11),
       'feature_names': ('0', '2', '3', '6', '7', '8', '9', '11')},
      9: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.88888889, 0.95555556, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 6, 7, 8, 9, 11),
       'feature_names': ('0', '1', '2', '3', '6', '7', '8', '9', '11')},
      10: {'avg_score': 0.9500000000000001,
       'cv_scores': array([0.86666667, 0.95555556, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 6, 7, 8, 9, 10, 11),
       'feature_names': ('0', '1', '2', '3', '6', '7', '8', '9', '10', '11')},
      11: {'avg_score': 0.9388888888888889,
       'cv_scores': array([0.84444444, 0.93333333, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11),
       'feature_names': ('0', '1', '2', '3', '5', '6', '7', '8', '9', '10', '11')},
      12: {'avg_score': 0.9653746770025839,
```

```
 12: {'avg_score': 0.8653746770025839,
  'cv_scores': array([0.82222222, 0.84444444, 0.91111111, 0.88372093]),
  'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11),
  'feature_names': ('0',
   '1',
   '2',
   '3',
   '4',
   '5',
   '6',
   '7',
   '8',
   '9',
   '10',
   '11')},
 13: {'avg_score': 0.6815245478036176,
  'cv_scores': array([0.6       , 0.68888889, 0.6       , 0.8372093 ]),
  'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
  'feature_names': ('0',
   '1',
   '2',
   '3',
   '4',
   '5',
   '6',
   '7',
   '8',
   '9',
   '10',
   '11',
   '12')}}
```

```
sbs.k_feature_idx_
```

```
(6,)
```

```
sbs.k_feature_names_
```

```
('6',)
```

```
sbs.k_score_
```

```
0.7532299741602068
```

Из проведённого анализа видно, что наивысшей точностью модель обладает при 7 признаках. Проведём повторный прогон, для нахождения 7 признаков.

```
sbs = SFS(knn,
          k_features=7,
          forward=False,
          floating=False,
```

```
          verbose=2,
          scoring='accuracy',
          cv=4)
sbs = sbs.fit(wine_X, wine_y)


sbs.subsets_

     {7: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.86666667, 0.95555556, 1.        , 1.        ]),
       'feature_idx': (0, 2, 3, 6, 8, 9, 11),
       'feature_names': ('0', '2', '3', '6', '8', '9', '11')},
      8: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.86666667, 0.95555556, 1.        , 1.        ]),
       'feature_idx': (0, 2, 3, 6, 7, 8, 9, 11),
       'feature_names': ('0', '2', '3', '6', '7', '8', '9', '11')},
      9: {'avg_score': 0.9555555555555556,
       'cv_scores': array([0.88888889, 0.95555556, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 6, 7, 8, 9, 11),
       'feature_names': ('0', '1', '2', '3', '6', '7', '8', '9', '11')},
      10: {'avg_score': 0.9500000000000001,
       'cv_scores': array([0.86666667, 0.95555556, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 6, 7, 8, 9, 10, 11),
       'feature_names': ('0', '1', '2', '3', '6', '7', '8', '9', '10', '11')},
      11: {'avg_score': 0.9388888888888889,
       'cv_scores': array([0.84444444, 0.93333333, 0.97777778, 1.        ]),
       'feature_idx': (0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11),
       'feature_names': ('0', '1', '2', '3', '5', '6', '7', '8', '9', '10', '11')},
      12: {'avg_score': 0.8653746770025839,
       'cv_scores': array([0.82222222, 0.84444444, 0.91111111, 0.88372093]),
       'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11),
       'feature_names': ('0',
        '1',
        '2',
        '3',
        '4',
        '5',
        '6',
        '7',
        '8',
        '9',
        '10',
        '11')},
      13: {'avg_score': 0.6815245478036176,
       'cv_scores': array([0.6       , 0.68888889, 0.6       , 0.8372093 ]),
       'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
       'feature_names': ('0',
        '1',
        '2',
        '3',
        '4',
        '5',
        '6',
        '7',
        '8',
```

```
        '9',
        '10',
        '11',
        '12')}}
```

Просмотрим индексы оставшихся 7 признаков

```
sbs.k_feature_idx_
```

```
    (0, 2, 3, 6, 8, 9, 11)
```

```
sbs.k_feature_names_
```

```
    ('0', '2', '3', '6', '8', '9', '11')
```

Посмотрим на точность полученной модели.

```
sbs.k_score_
```

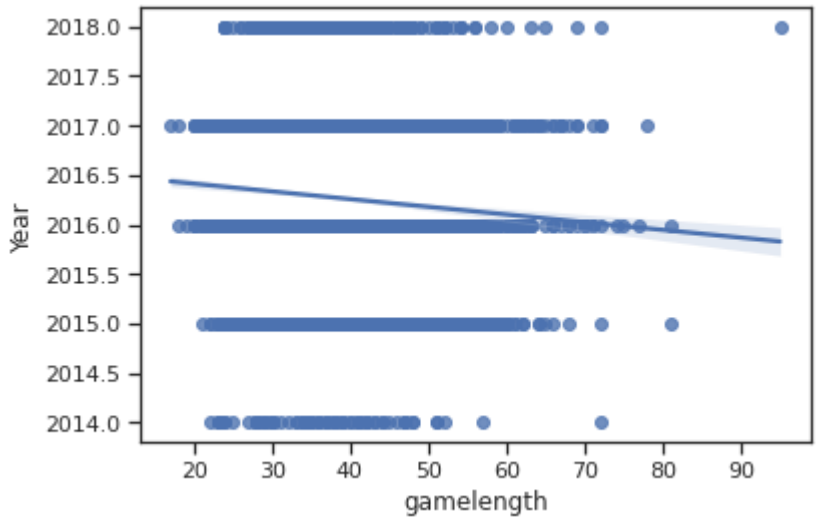
```
    0.9555555555555556
```

Дополнительное задание

Для пары произвольных колонок данных построить график "Диаграмма рассеяния".

```
sns.regplot(x=data['gamelength'], y=data['Year'])
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f91f0424b90>
```



Построили график рассеяния, показывающий зависимость между двумя признаками: gamelength (продолжительностью игры) и Year (годом)