МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЁТ

Лабораторная работа №2
по курсу «**Методы машинного обучения**»

ИСПОЛНИТЕЛИ: Елизаров О.О.
ФИО
группа ИУ5-21М _____
подпись

"__"_____2022 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.У.
ФИО
_____
подпись

"__"_____2022 г.

Москва - 2022

Обработка признаков

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
    1. устранение пропусков в данных;
    2. кодирование категориальных признаков;
    3. нормализацию числовых признаков.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
data = pd.read_csv('archive/Pokemon.csv')
data
```

Out[3]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Genera
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	
...	...	...	...	...	...	...	...	...	...	...	...	
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	

800 rows × 13 columns

In [5]:

```
data.shape
```

Out[5]:

```
(800, 13)
```

In [6]:

```
data.isnull().sum()
```

Out[6]:

```
#               0
Name            0
Type 1          0
Type 2        386
Total           0
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

In [13]:

```
[(c, data[c].isnull().mean()) for c in data.columns]
```

Out[13]:

```
[('#', 0.0),
 ('Name', 0.0),
 ('Type 1', 0.0),
 ('Type 2', 0.4825),
 ('Total', 0.0),
 ('HP', 0.0),
 ('Attack', 0.0),
 ('Defense', 0.0),
 ('Sp. Atk', 0.0),
 ('Sp. Def', 0.0),
 ('Speed', 0.0),
 ('Generation', 0.0),
 ('Legendary', 0.0)]
```

Слишком большой процент строк имеет пропуск, удалять строки опасно. Можно удалить колонку или засетить значение дефолтным(Normal)

In [22]:

```
data_clean = res = data.dropna(axis=1, how='any')
data_clean.shape
```

Out[22]:

```
(800, 12)
```

In [29]:

```
data_normal = data.fillna(value='Normal')
data_normal.isnull().sum()
```

Out[29]:

```
#             0
Name          0
Type 1        0
Type 2        0
Total         0
HP            0
Attack        0
Defense       0
Sp. Atk       0
Sp. Def       0
Speed         0
Generation    0
Legendary     0
dtype: int64
```

In [33]:

```
data_clean.dtypes
```

Out[33]:

```
#              int64
Name          object
Type 1        object
Total          int64
HP             int64
Attack         int64
Defense        int64
Sp. Atk        int64
Sp. Def        int64
Speed          int64
Generation     int64
Legendary       bool
dtype: object
```

Перейдем к стадии кодирования категориальных признаков. Попробуем кодировать тип покемонов.

In [53]:

```
from sklearn.preprocessing import LabelEncoder
# Label Encoding
```

In [54]:

```
le = LabelEncoder()
encoded_types = le.fit_transform(data_clean['Type 1'])
```

In [55]:

```
data_clean['Type 1'].unique()
```

Out[55]:

```
array(['Grass', 'Fire', 'Water', 'Bug', 'Normal', 'Poison', 'Electric',
       'Ground', 'Fairy', 'Fighting', 'Psychic', 'Rock', 'Ghost', 'Ice',
       'Dragon', 'Dark', 'Steel', 'Flying'], dtype=object)
```

In [56]:

```
np.unique(encoded_types)
```

Out[56]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17])
```

In [57]:

```
le.inverse_transform([0, 1, 2, 3])
```

Out[57]:

```
array(['Bug', 'Dark', 'Dragon', 'Electric'], dtype=object)
```

One-hot encoding

In [58]:

```
pd.get_dummies(data_clean['Type 1']).head()
```

Out[58]:

	Bug	Dark	Dragon	Electric	Fairy	Fighting	Fire	Flying	Ghost	Grass	Ground	Ice	Norr
0	0	0	0	0	0	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	0	1	0	0	
2	0	0	0	0	0	0	0	0	0	1	0	0	
3	0	0	0	0	0	0	0	0	0	1	0	0	
4	0	0	0	0	0	0	1	0	0	0	0	0	

In [ ]:

In [59]:

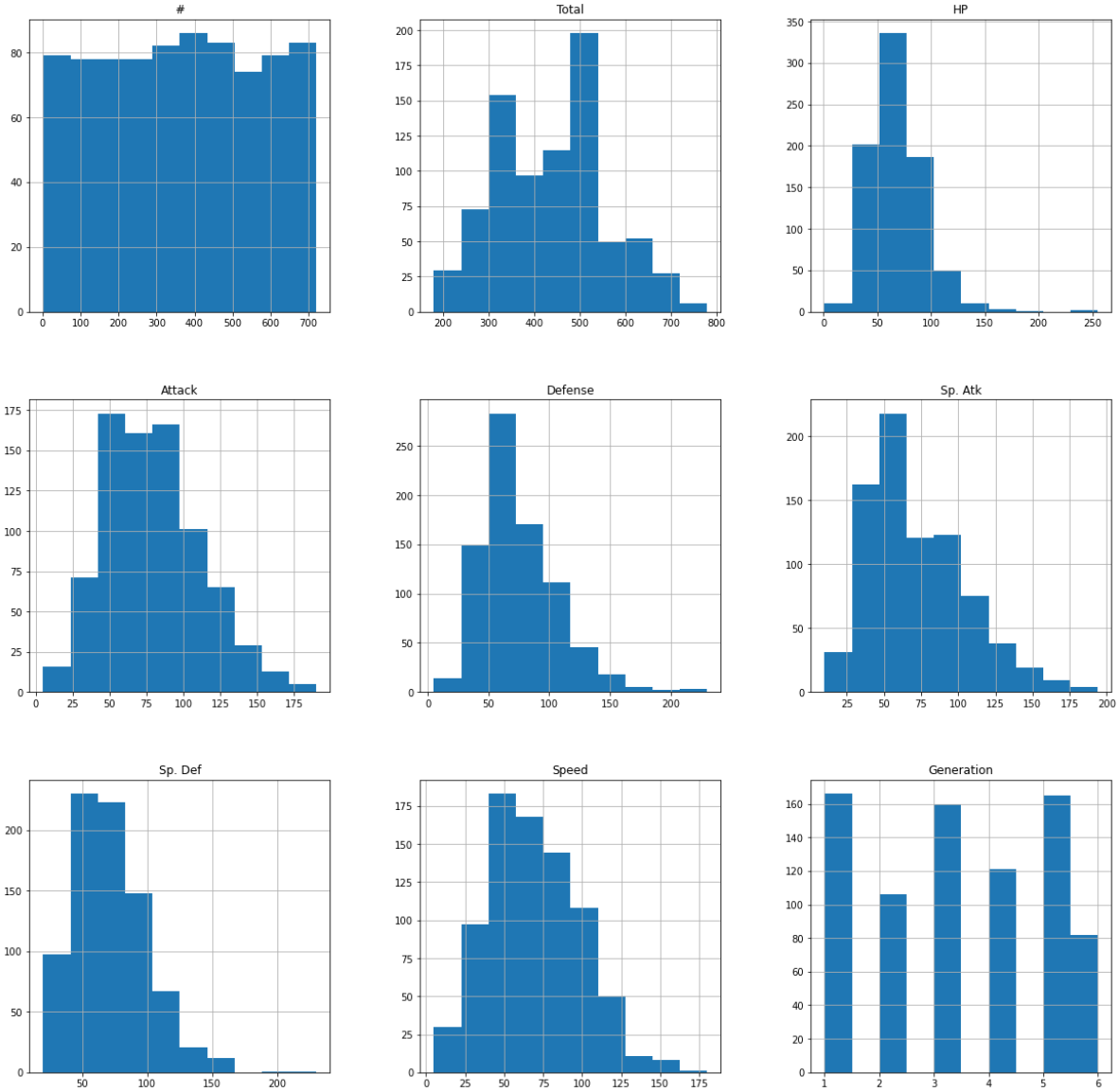
```
data_clean.groupby("Type 1").mean().sort_values("Total")
```

Out[59]:

Type 1	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	
Bug	334.492754	378.927536	56.884058	70.971014	70.724638	53.869565	64.797101	61
Poison	251.785714	399.142857	67.250000	74.678571	68.821429	60.428571	64.392857	63.
Normal	319.173469	401.683673	77.275510	73.469388	59.846939	55.816327	63.724490	71.
Fairy	449.529412	413.176471	74.117647	61.529412	65.705882	78.529412	84.705882	48.
Fighting	363.851852	416.444444	69.851852	96.777778	65.925926	53.111111	64.703704	66.
Grass	344.871429	421.142857	67.271429	73.214286	70.800000	77.500000	70.428571	61.
Water	303.089286	430.455357	72.062500	74.151786	72.946429	74.812500	70.517857	65.
Ice	423.541667	433.458333	72.000000	72.750000	71.416667	77.541667	76.291667	63.
Ground	356.281250	437.500000	73.781250	95.750000	84.843750	56.468750	62.750000	63.
Ghost	486.500000	439.562500	64.437500	73.781250	81.187500	79.343750	76.468750	64.
Electric	363.500000	443.409091	59.795455	69.090909	66.295455	90.022727	73.704545	84.
Dark	461.354839	445.741935	66.806452	88.387097	70.225806	74.645161	69.516129	76.
Rock	392.727273	453.750000	65.363636	92.863636	100.795455	63.340909	75.477273	55.
Fire	327.403846	458.076923	69.903846	84.769231	67.769231	88.980769	72.211538	74.
Psychic	380.807018	475.947368	70.631579	71.456140	67.684211	98.403509	86.280702	81.
Flying	677.750000	485.000000	70.750000	78.750000	66.250000	94.250000	72.500000	102.
Steel	442.851852	487.703704	65.222222	92.703704	126.370370	67.518519	80.629630	55.
Dragon	474.375000	550.531250	83.312500	112.125000	86.375000	96.843750	88.843750	83.

In [61]:

```
data_clean.hist(figsize=(20,20))
plt.show()
```



In [69]:

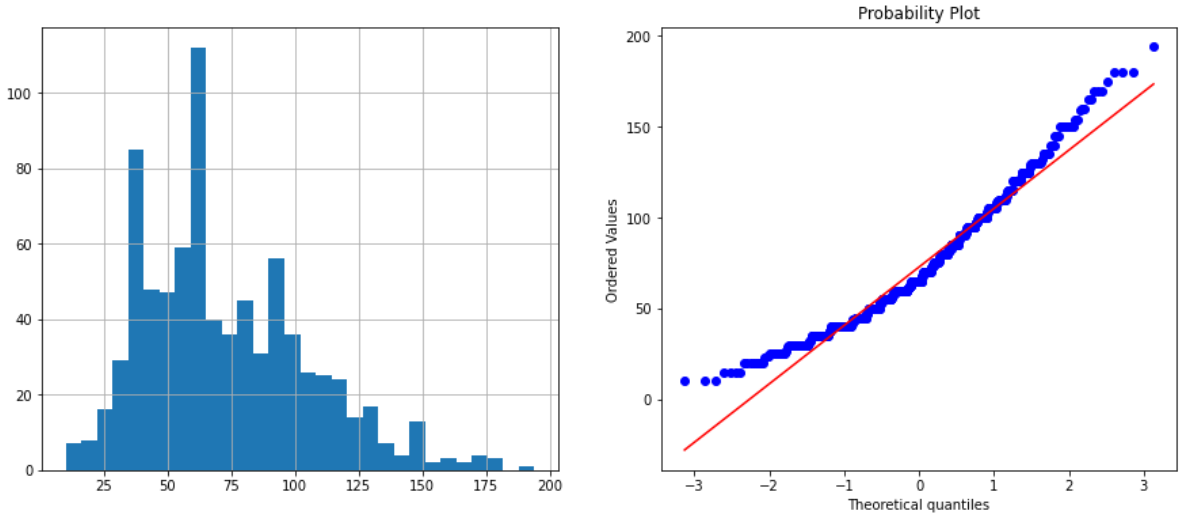
```
import scipy.stats as stats

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

def diagnostic(dat):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    dat.hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(dat, dist="norm", plot=plt)
    plt.show()
```

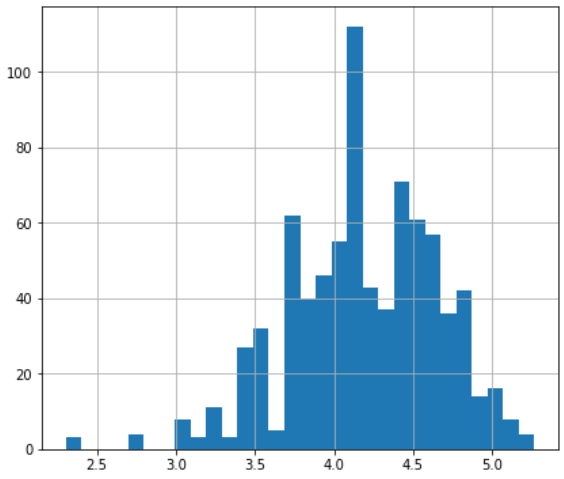
In [70]:

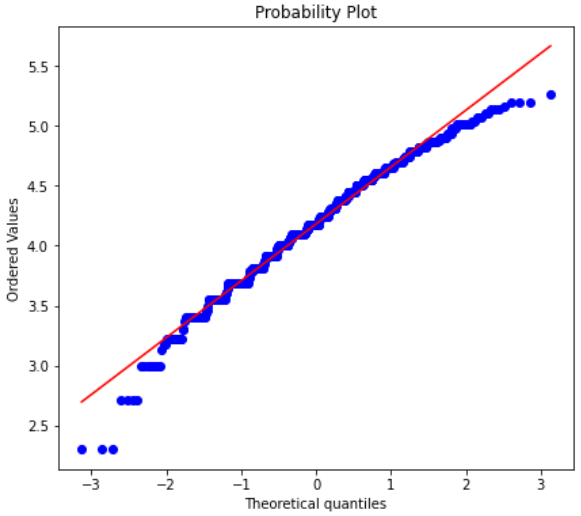
```
diagnostic_plots(data_clean, 'Sp. Atk')
```



In [73]:

```
diagnostic(np.log(data_clean['Sp. Atk']))
```

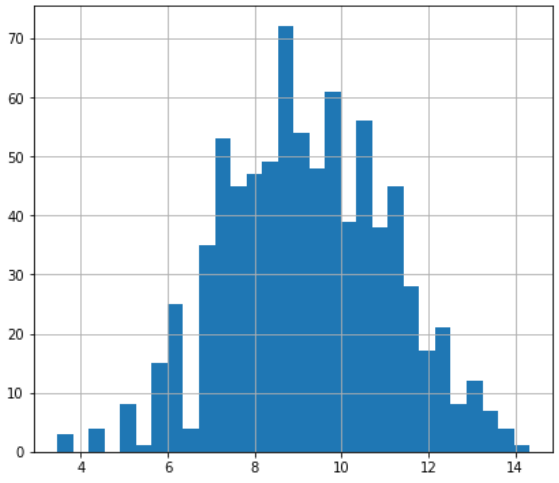


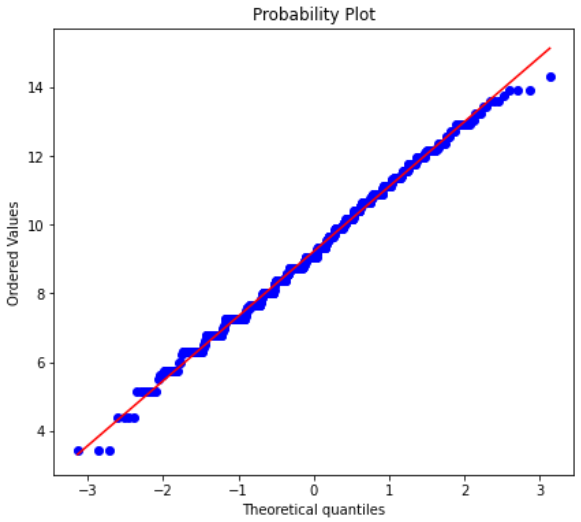


In [75]:

```
data_clean['Box'], param = stats.boxcox(data_clean['Sp. Atk'])
print(param)
diagnostic_plots(data_clean, 'Box')
```

```
0.33253199847700005
```





Преобразование Бокса-Кокса показывает и правда лучший результат, как и в лекции. РАспределение и правда почти нормальным

In [ ]:

```

```

In [ ]:

```

```