

Лабораторная работа №2  
по дисциплине  
«Методы машинного обучения»  
на тему  
«Обработка признаков»

Выполнила:  
студентка группы ИУ5-21 М  
Базанова А.Г.

---

# 1. Лабораторная №2

**Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.**

**Задание:**

1. Выбрать набор данных (датасет), содержащий категориальные 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.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
  - устранение пропусков в данных;
  - кодирование категориальных признаков;
  - нормализация числовых признаков.

## 2. Описание данных

- Date - Дата наблюдений
- Location - Название локации, в которой расположена метеорологическая станция
- MinTemp - Минимальная температура в градусах цельсия
- MaxTemp - Максимальная температура в градусах цельсия
- Rainfall - Количество осадков, зафиксированных за день в мм
- Evaporation - Так называемое “pan evaporation” класса A (мм) за 24 часа до 9 утра
- Sunshine - Число солнечных часов за день
- WindGustDir - направление самого сильного порыва ветра за последние 24 часа
- WindGustSpeed - скорость (км / ч) самого сильного порыва ветра за последние 24 часа
- WindDir9am - направление ветра в 9 утра

```
[3]: import sklearn
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats

import matplotlib.pyplot as plt
```

```
[67]: data = pd.read_csv('weatherAUS.csv', parse_dates=['Date'])
```

```
[3]: data.head()
```

```
[3]:
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	\
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	

	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	\
0	W	44.0	W	...	22.0	1007.7	
1	WNW	44.0	NNW	...	25.0	1010.6	
2	WSW	46.0	W	...	30.0	1007.6	
3	NE	24.0	SE	...	16.0	1017.6	
4	W	41.0	ENE	...	33.0	1010.8	

	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RISK_MM	\
0	1007.1	8.0	NaN	16.9	21.8	No	0.0	
1	1007.8	NaN	NaN	17.2	24.3	No	0.0	
2	1008.7	NaN	2.0	21.0	23.2	No	0.0	
3	1012.8	NaN	NaN	18.1	26.5	No	1.0	
4	1006.0	7.0	8.0	17.8	29.7	No	0.2	

	RainTomorrow
0	No
1	No
2	No
3	No
4	No

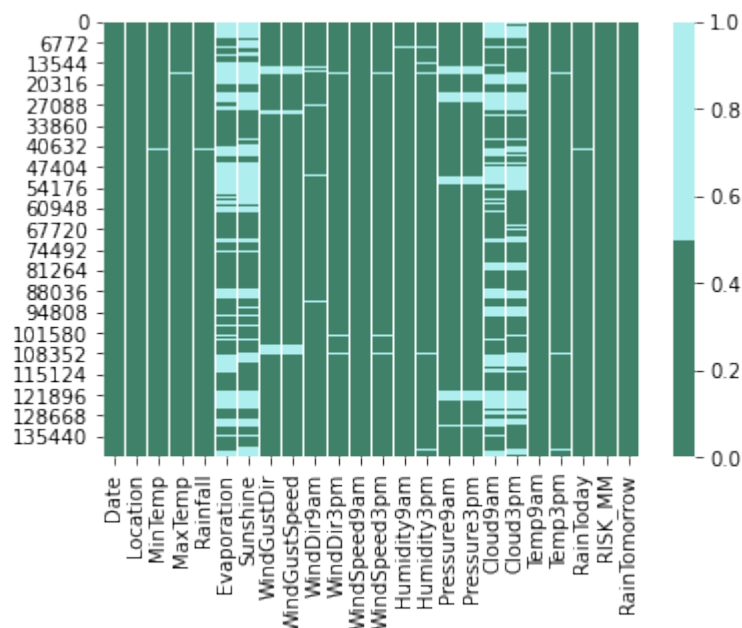
[5 rows x 24 columns]

## 2.1. Устранение пропусков в данных

Светлым цветом отметим пропущенные данные

```
[4]: cols = data.columns
     colours = ['#408169', '#AFEEEE']
     sns.heatmap(data[cols].isnull(), cmap=sns.color_palette(colours))
```

```
[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe232f67f40>
```



```
[63]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Date                  142193 non-null  datetime64[ns]
1   Location              142193 non-null  object
2   MinTemp               141556 non-null  float64
3   MaxTemp               141871 non-null  float64
4   Rainfall              140787 non-null  float64
5   Evaporation           81350 non-null   float64
6   Sunshine              74377 non-null   float64
7   WindGustDir           132863 non-null  object
8   WindGustSpeed         132923 non-null  float64
9   WindDir9am            132180 non-null  object
10  WindDir3pm            138415 non-null  object
11  WindSpeed9am          140845 non-null  float64
12  WindSpeed3pm          139563 non-null  float64
13  Humidity9am           140419 non-null  float64
14  Humidity3pm           138583 non-null  float64
15  Pressure9am           128179 non-null  float64
16  Pressure3pm           128212 non-null  float64
17  Cloud9am              88536 non-null   float64
18  Cloud3pm              85099 non-null   float64
19  Temp9am               141289 non-null  float64
20  Temp3pm               139467 non-null  float64
21  RainToday             140787 non-null  object
22  RISK_MM               142193 non-null  float64
23  RainTomorrow          142193 non-null  object
dtypes: datetime64[ns](1), float64(17), object(6)
memory usage: 26.0+ MB
```

Рассмотрим числовые колонки с пропущенными значениями

```
[68]: total_count = data.shape[0]
num_cols = []
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('    {}.    {}.    {}, {}%.'.format(col, dt,
temp_null_count, temp_perc))
```

```
MinTemp.    float64.    637, 0.45%.
MaxTemp.    float64.    322, 0.23%.
```

Rainfall.	float64.	1406, 0.99%.
Evaporation.	float64.	60843,
42.79%.		
Sunshine.	float64.	67816, 47.69%.
WindGustSpeed.	float64.	9270,
6.52%.		
WindSpeed9am.	float64.	1348,
0.95%.		
WindSpeed3pm.	float64.	2630,
1.85%.		
Humidity9am.	float64.	1774, 1.25%.
Humidity3pm.	float64.	3610, 2.54%.
Pressure9am.	float64.	14014,
9.86%.		
Pressure3pm.	float64.	13981,
9.83%.		
Cloud9am.	float64.	53657, 37.74%.
Cloud3pm.	float64.	57094, 40.15%.
Temp9am.	float64.	904, 0.64%.
Temp3pm.	float64.	2726, 1.92%.

Фильтр по колонкам с пропущенными значениями

```
[6]: data_num = data[num_cols]
data_num
```

```
[6]:
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	\
0	13.4	22.9	0.6	NaN	NaN	44.0	
1	7.4	25.1	0.0	NaN	NaN	44.0	
2	12.9	25.7	0.0	NaN	NaN	46.0	
3	9.2	28.0	0.0	NaN	NaN	24.0	
4	17.5	32.3	1.0	NaN	NaN	41.0	
...	...	...	...	...	...	...	
142188	3.5	21.8	0.0	NaN	NaN	31.0	
142189	2.8	23.4	0.0	NaN	NaN	31.0	
142190	3.6	25.3	0.0	NaN	NaN	22.0	
142191	5.4	26.9	0.0	NaN	NaN	37.0	
142192	7.8	27.0	0.0	NaN	NaN	28.0	

	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	\
0	20.0	24.0	71.0	22.0	1007.7	
1	4.0	22.0	44.0	25.0	1010.6	
2	19.0	26.0	38.0	30.0	1007.6	
3	11.0	9.0	45.0	16.0	1017.6	
4	7.0	20.0	82.0	33.0	1010.8	
...	...	...	...	...	...	
142188	15.0	13.0	59.0	27.0	1024.7	
142189	13.0	11.0	51.0	24.0	1024.6	
142190	13.0	9.0	56.0	21.0	1023.5	
142191	9.0	9.0	53.0	24.0	1021.0	
142192	13.0	7.0	51.0	24.0	1019.4	

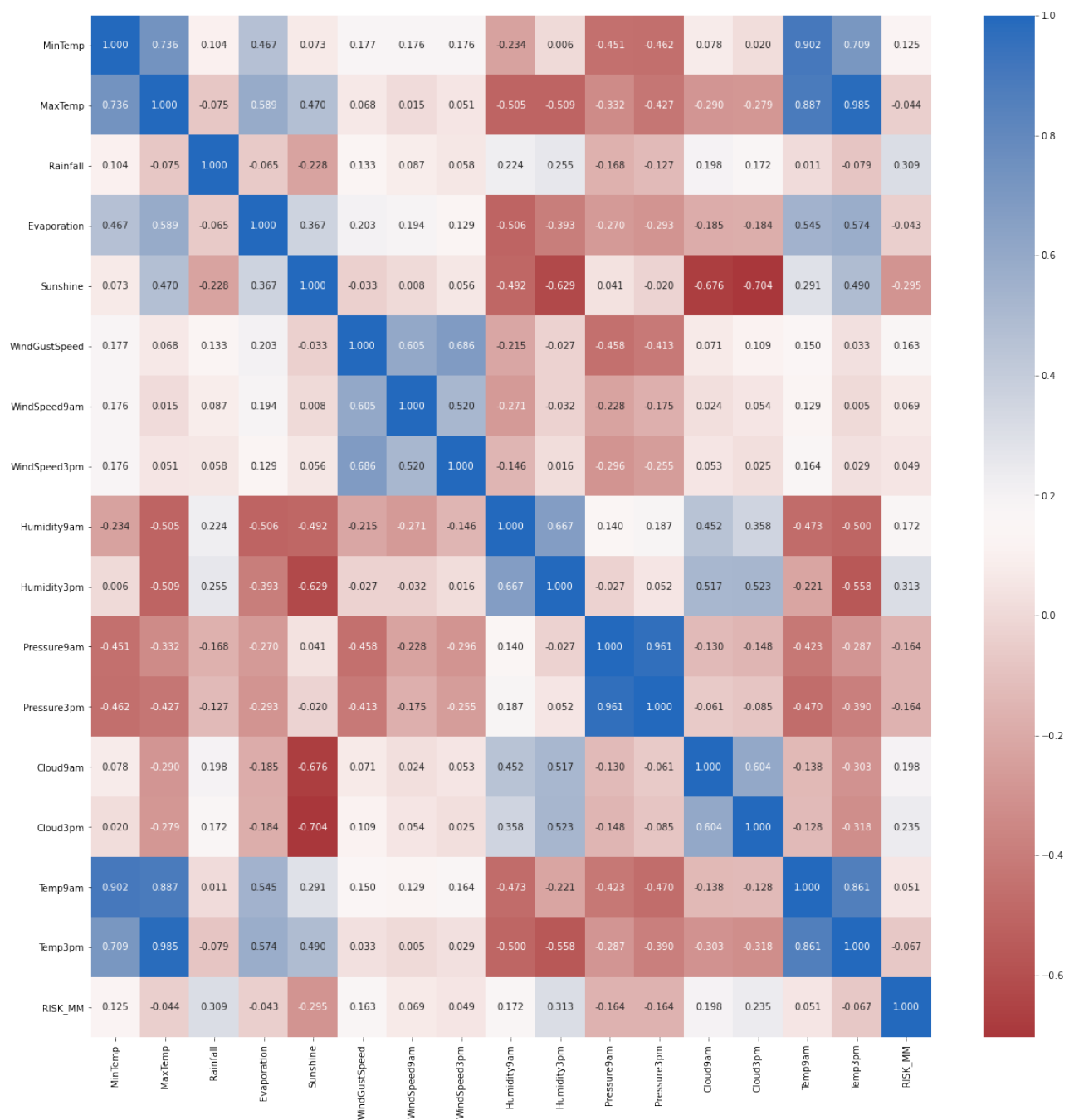
	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
0	1007.1	8.0	NaN	16.9	21.8
1	1007.8	NaN	NaN	17.2	24.3
2	1008.7	NaN	2.0	21.0	23.2
3	1012.8	NaN	NaN	18.1	26.5
4	1006.0	7.0	8.0	17.8	29.7
...	...	...	...	...	...
142188	1021.2	NaN	NaN	9.4	20.9
142189	1020.3	NaN	NaN	10.1	22.4
142190	1019.1	NaN	NaN	10.9	24.5
142191	1016.8	NaN	NaN	12.5	26.1
142192	1016.5	3.0	2.0	15.1	26.0

[142193 rows x 16 columns]

```
[30]: import sys
import numpy
numpy.set_printoptions(threshold=sys.maxsize)

corrmat = data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corrmat, annot=True, fmt='.3f', cmap="vlag_r")
```

[30]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fe22c411130>



В столбце Evaporation 42.79% пропущенных данных, и его корреляция с целевым признаком низкая, так что легче всего этот столбец удалить.

```
[69]: data = data.drop(['Evaporation'], axis = 1)
      num_cols.remove('Evaporation')
```

Все распределения, кроме Sunshine, одномодальные, так что будем использовать для заполнения пропусков моду. Для Sunshine медиану.

```
[70]: data['Sunshine'] = data['Sunshine'].fillna(data.median(numeric_only=True))
```

```
[71]: data['Humidity9am'] = data['Humidity9am'].fillna(data['Humidity9am'].mode())
```

```
[72]: data = data.fillna(data.mode())
```

Рассмотрим пропуски в категориальных данных

```
[21]: cat_cols = []
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('      {}. {}      {}, {}%.'.format(col, dt,
↪temp_null_count, temp_perc))
```

WindGustDir.	object.	9330, 6.56%.
WindDir9am.	object.	10013, 7.04%.
WindDir3pm.	object.	3778, 2.66%.
RainToday.	object.	1406, 0.99%.

```
[75]: for col in data[cat_cols]:
      print('      {}. {}'.format(col, data[col].unique()))
```

```
WindGustDir. ['W' 'WNW' 'WSW' 'NE' 'NNW' 'N' 'NNE' 'SW' 'ENE' 'SSE' 'S'
'NW' 'SE' 'ESE'
nan 'E' 'SSW']
WindDir9am. ['W' 'NNW' 'SE' 'ENE' 'SW' 'SSE' 'S' 'NE' nan 'SSW' 'N'
'WSW' 'ESE' 'E'
'NW' 'WNW' 'NNE']
WindDir3pm. ['WNW' 'WSW' 'E' 'NW' 'W' 'SSE' 'ESE' 'ENE' 'NNW' 'SSW' 'SW'
'SE' 'N' 'S'
'NNE' nan 'NE']
RainToday. ['No' 'Yes' nan]
```

```
[73]: data[:] = SimpleImputer(missing_values=np.nan, strategy='most_frequent').
↪fit_transform(data)
```

```
[110]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 23 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Date                  142193 non-null  datetime64[ns]
1   Location              142193 non-null  object
2   MinTemp               142193 non-null  float64
3   MaxTemp               142193 non-null  float64
4   Rainfall              142193 non-null  float64
5   Sunshine              142193 non-null  float64
6   WindGustDir           142193 non-null  object
7   WindGustSpeed         142193 non-null  float64
```



```

8   WindDir9am      142193 non-null object
9   WindDir3pm      142193 non-null object
10  WindSpeed9am     142193 non-null float64
11  WindSpeed3pm     142193 non-null float64
12  Humidity9am      142193 non-null float64
13  Humidity3pm      142193 non-null float64
14  Pressure9am      142193 non-null float64
15  Pressure3pm      142193 non-null float64
16  Cloud9am         142193 non-null float64
17  Cloud3pm         142193 non-null float64
18  Temp9am          142193 non-null float64
19  Temp3pm          142193 non-null float64
20  RainToday        142193 non-null object
21  RISK_MM          142193 non-null float64
22  RainTomorrow     142193 non-null object
dtypes: datetime64[ns](1), float64(16), object(6)
memory usage: 25.0+ MB

```

```
[42]: data.isnull().sum()
```

```

[42]: Date          0
      Location      0
      MinTemp       0
      MaxTemp       0
      Rainfall      0
      Sunshine      0
      WindGustDir    0
      WindGustSpeed  0
      WindDir9am     0
      WindDir3pm     0
      WindSpeed9am   0
      WindSpeed3pm   0
      Humidity9am    0
      Humidity3pm    0
      Pressure9am    0
      Pressure3pm    0
      Cloud9am       0
      Cloud3pm       0
      Temp9am        0
      Temp3pm        0
      RainToday      0
      RISK_MM        0
      RainTomorrow   0
      dtype: int64

```

## 2.2. Кодирование категориальных признаков

```
[74]: data['RainToday'] = data['RainToday'].apply(lambda x: 1 if x == 'Yes' else 0)
data['RainTomorrow'] = data['RainTomorrow'].apply(lambda x: 1 if x == 'Yes' else 0)
```

```
[136]: data['Location'].unique()
```

```
[136]: array(['Albury', 'BadgerysCreek', 'Cobar', 'CoffsHarbour', 'Moree',
        'Newcastle', 'NorahHead', 'NorfolkIsland', 'Penrith', 'Richmond',
        'Sydney', 'SydneyAirport', 'WaggaWagga', 'Williamtown',
        'Wollongong', 'Canberra', 'Tuggeranong', 'MountGinini', 'Ballarat',
        'Bendigo', 'Sale', 'MelbourneAirport', 'Melbourne', 'Mildura',
        'Nhil', 'Portland', 'Watsonia', 'Dartmoor', 'Brisbane', 'Cairns',
        'GoldCoast', 'Townsville', 'Adelaide', 'MountGambier', 'Nuriootpa',
        'Woomera', 'Albany', 'Witchcliffe', 'PearceRAAF', 'PerthAirport',
        'Perth', 'SalmonGums', 'Walpole', 'Hobart', 'Launceston',
        'AliceSprings', 'Darwin', 'Katherine', 'Uluru'], dtype=object)
```

Слишком много категорий Location для OneHotEncoder

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

le = LabelEncoder()
data['Location'] = le.fit_transform(data['Location'])
```

```
[76]: categorical = ['WindDir3pm', 'WindDir9am', 'WindGustDir']

data = pd.concat([data, pd.get_dummies(data[categorical],
    columns=categorical, drop_first=True)], axis=1)
data.drop(categorical, axis=1, inplace=True)
```

```
[11]: data.shape
```

```
[11]: (142193, 65)
```

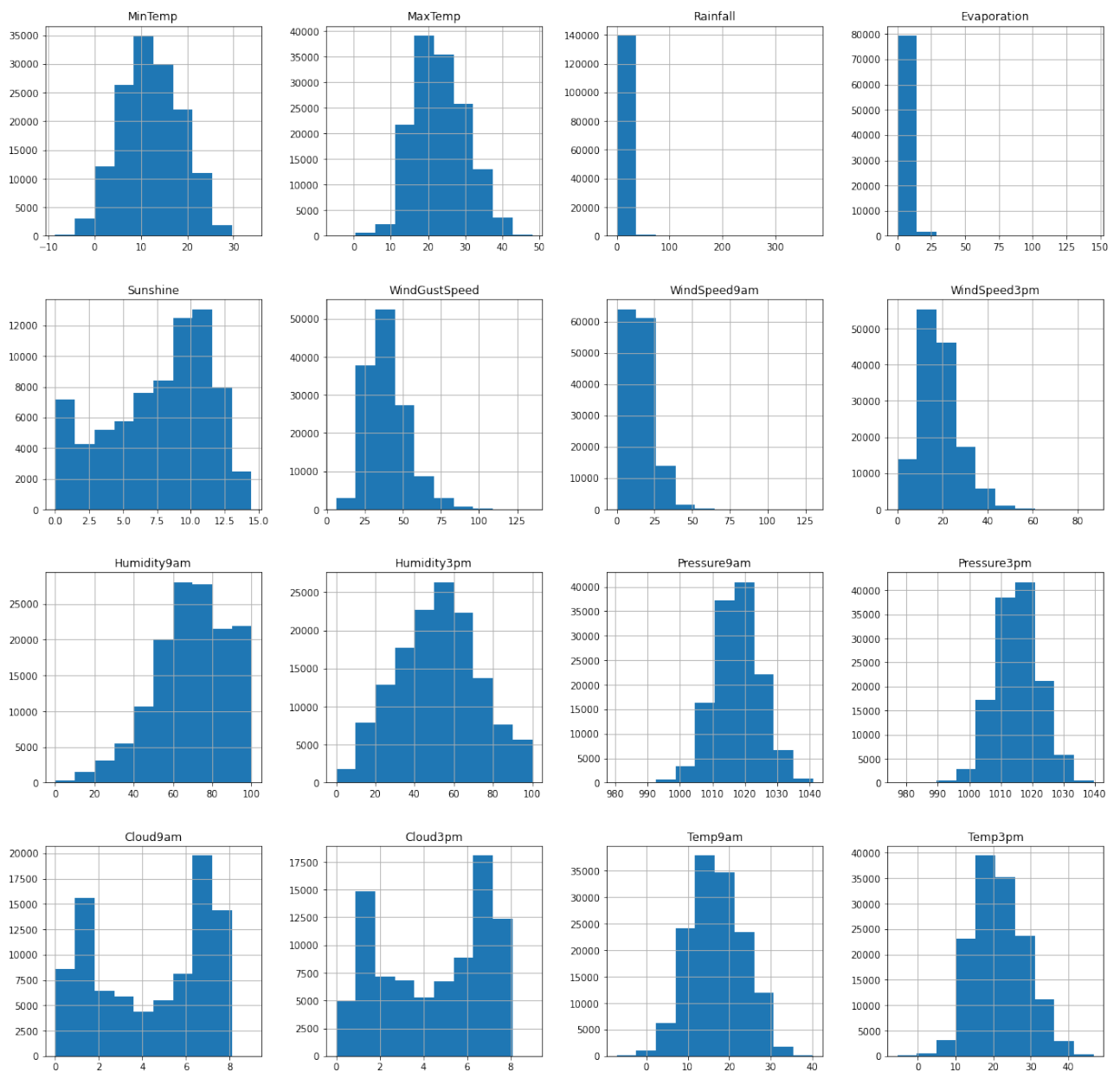
## 2.3. Нормализация числовых признаков

```
[77]: data[num_cols] = data[data[num_cols] > 0][num_cols]
```

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

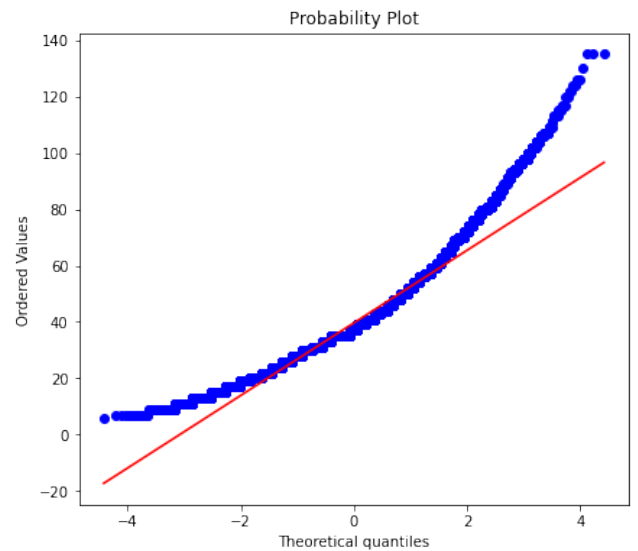
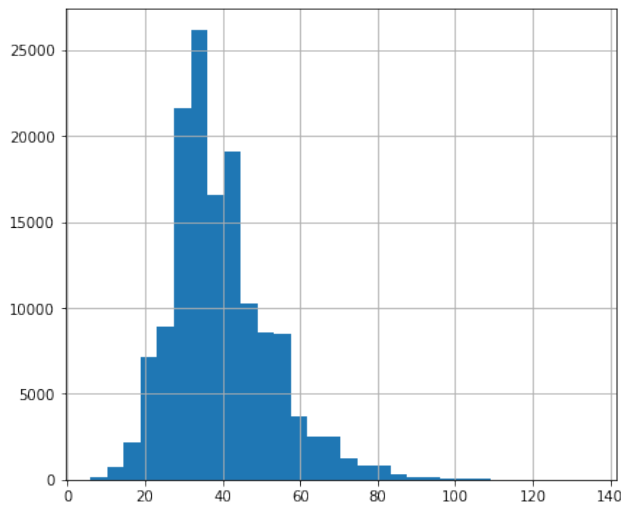
```
[128]: def diagnostic_plots_data(df):
plt.figure(figsize=(15,6))
#
plt.subplot(1, 2, 1)
df.hist(bins=30)
## Q-Q plot
plt.subplot(1, 2, 2)
stats.probplot(df, dist="norm", plot=plt)
plt.show()
```

```
[59]: data_num.hist(figsize=(20,20))
plt.show()
```

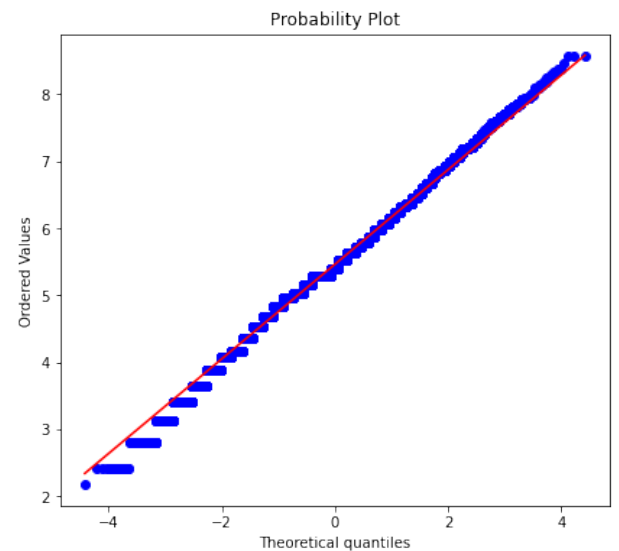
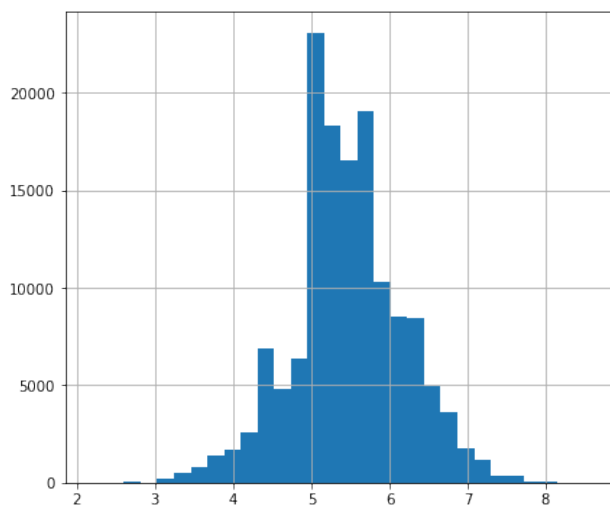


```
[138]: diagnostic_plots(data, 'WindGustSpeed')

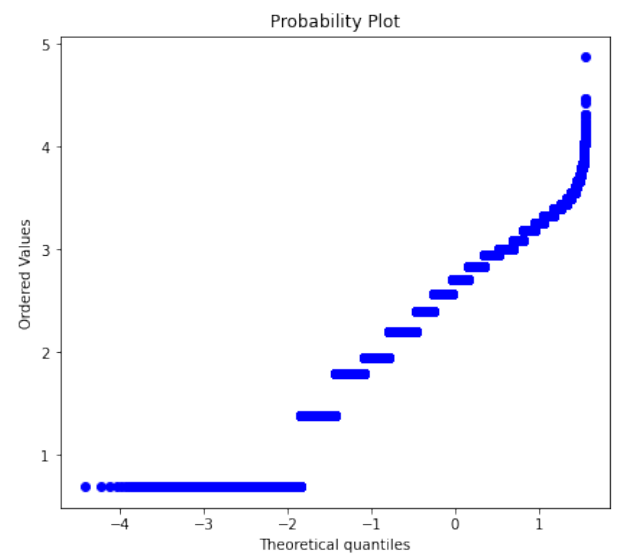
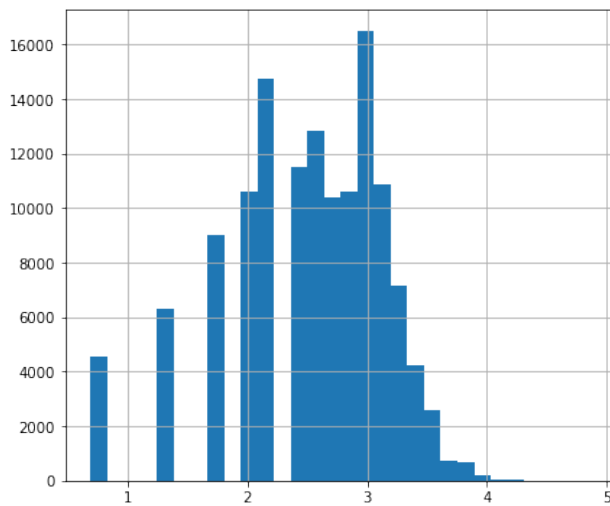
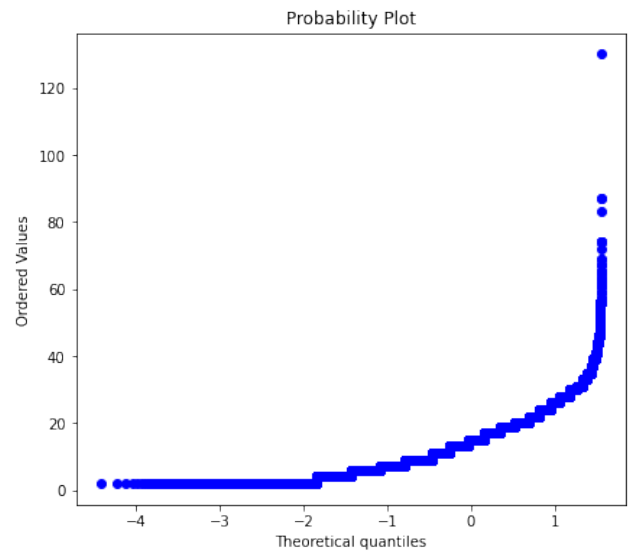
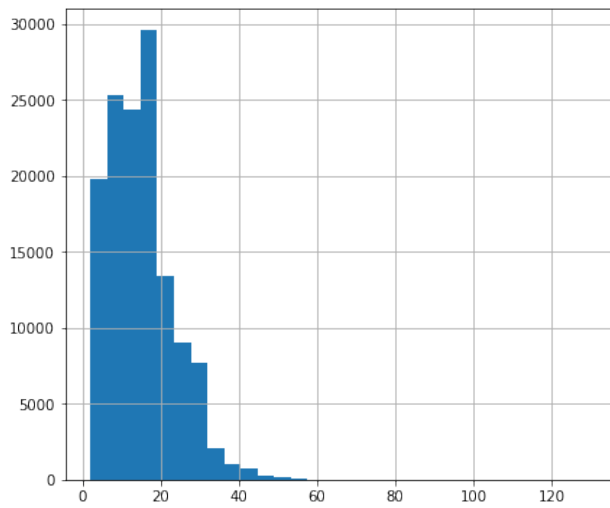
data['WindGustSpeed_boxcox'], param = stats.boxcox(data['WindGustSpeed'])
print('          = {}'.format(param))
diagnostic_plots(data, 'WindGustSpeed_boxcox')
```



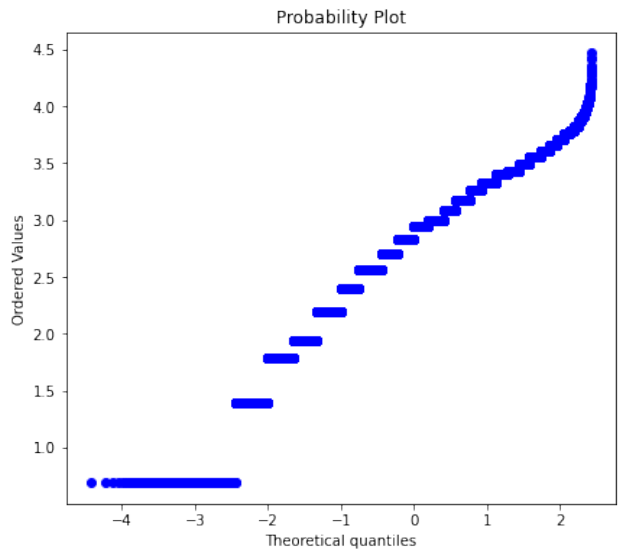
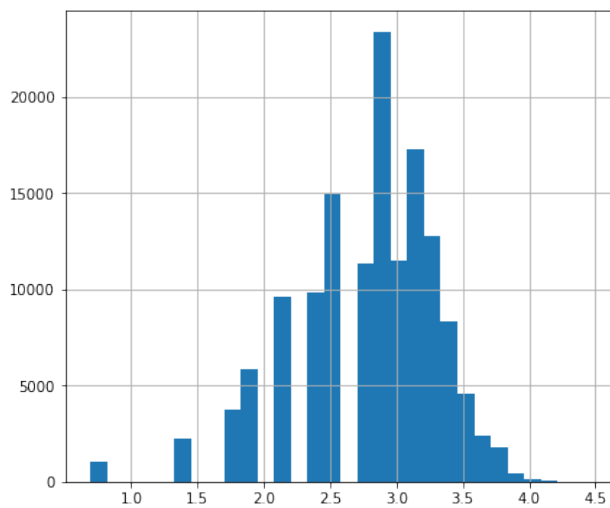
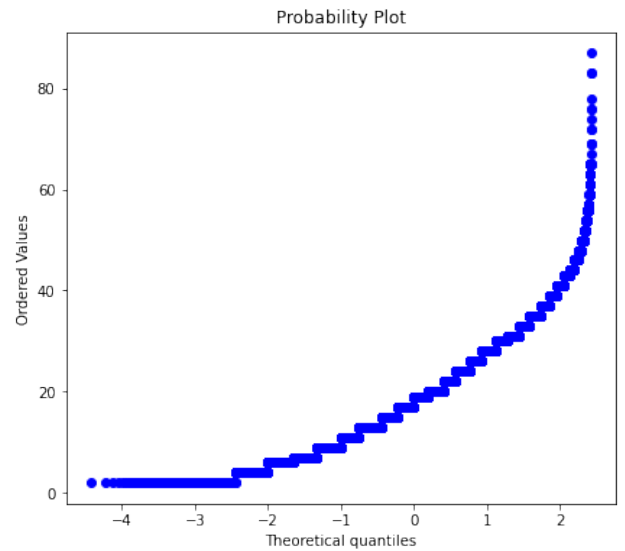
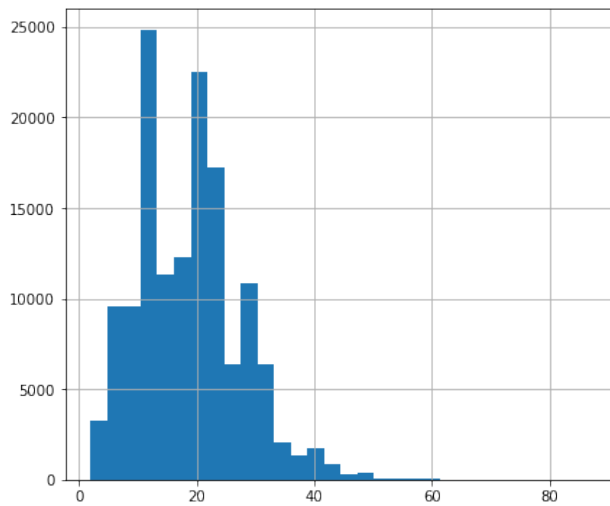
= 0.21000467599892894



```
[139]: diagnostic_plots(data, 'WindSpeed9am')
diagnostic_plots_data( np.log(data['WindSpeed9am']) )
```



```
[167]: diagnostic_plots(data, 'WindSpeed3pm')
        diagnostic_plots_data( np.log( (data['WindSpeed3pm']) ) ) )
```



```
[81]: data['WindSpeed9am'] = np.log( (data['WindSpeed9am']) )
      data['WindSpeed3pm'] = np.log( (data['WindSpeed3pm']) )
      data['WindGustSpeed'], param = stats.boxcox(data['WindGustSpeed'])
```