Савченко Г.А. ИУ5-21М

Вариант 14

Каждая задача предполагает использование набора данных.

Набор данных выбирается Вами произвольно с учетом следующих условий:

- Вы можете использовать один набор данных для решения всех задач, или решать каждую задачу на своем наборе данных.
- Набор данных должен отличаться от набора данных, который использовался в лекции для решения рассматриваемой задачи.
- Вы можете выбрать произвольный набор данных (например тот, который Вы использовали в лабораторных работах) или создать собственный набор данных (что актуально для некоторых задач, например, для задач удаления псевдоконстантных или повторяющихся признаков).
- Выбранный или созданный Вами набор данных должен удовлетворять условиям поставленной задачи. Например, если решается задача устранения пропусков, то набор данных должен содержать пропуски.

Номер задачи №1 - 14

Номер задачи №2 - 34

Задача №14.

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "квадратный корень"

Задача №34.

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод вложений (embedded method). Используйте подход на основе линейной или логистической регрессии (в зависимости от того, на решение какой задачи ориентирован выбранный Вами набор данных - задачи регрессии или задачи классификации).

Дополнительные требования:

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

Загрузка и первичный анализ данных

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('bike-hour.csv', sep=",")
```

In [3]:
```
data.head(5)
```

Out[3]:

	instant	dteday	season	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hu
0	1	01-01-2011	1	1	0	0	6	0	1	0.24	0.2879	0.
1	2	01-01-2011	1	1	1	0	6	0	1	0.22	0.2727	0.
2	3	01-01-2011	1	1	2	0	6	0	1	0.22	0.2727	0.
3	4	01-01-2011	1	1	3	0	6	0	1	0.24	0.2879	0.
4	5	01-01-2011	1	1	4	0	6	0	1	0.24	0.2879	0.

Датасет

Информация об атрибутах:

- instant: индекс записи
- dteday: дата
- season: Сезон (1: зима, 2: весна, 3: лето, 4: осень)
- mnth: месяц (от 1 до 12)
- hour: час (от 0 до 23)
- holiday: выходной или нет
- weekday: день недели
- workingday: если день не является ни выходным, ни праздничным - 1, в противном случае - 0.
- weathersit:

  1: Ясно, Небольшая облачность, Небольшая облачность,

  2: Туман + Облачно, Туман + Разбитые облака, Туман + Несколько облаков, Туман

  3: слабый снег, легкий дождь + гроза + рассеянные облака, легкий дождь + рассеянные облака

4: сильный дождь + ледяные поддоны + гроза + туман, снег + туман

- temp: нормализованная температура в градусах Цельсия
- atemp: нормализованная температура ощущения в градусах Цельсия
- hum: нормализованная влажность
- windspeed: нормализованная скорость ветра
- casul: количество случайных прохожих
- cnt: общее количество взятых напрокат велосипедов

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8645 entries, 0 to 8644
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     8645 non-null   int64  
 1   dteday      8645 non-null   object 
 2   season      8645 non-null   int64  
 3   mnth        8645 non-null   int64  
 4   hr          8645 non-null   int64  
 5   holiday     8645 non-null   int64  
 6   weekday     8645 non-null   int64  
 7   workingday  8645 non-null   int64  
 8   weathersit  8645 non-null   int64  
 9   temp        8645 non-null   float64
 10  atemp       8645 non-null   float64
 11  hum         8645 non-null   float64
 12  windspeed   8645 non-null   float64
 13  casual      8645 non-null   int64  
 14  cnt         8645 non-null   int64  
dtypes: float64(4), int64(10), object(1)
memory usage: 1013.2+ KB
```

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

Out[5]:

	instant	season	mnth	hr	holiday	weekday	workingday	
count	8645.000000	8645.000000	8645.000000	8645.000000	8645.000000	8645.000000	8645.000000	8
mean	4323.000000	2.513592	6.573973	11.573626	0.027646	3.012724	0.683748	
std	2495.740872	1.105477	3.428147	6.907822	0.163966	2.006370	0.465040	
min	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	2162.000000	2.000000	4.000000	6.000000	0.000000	1.000000	0.000000	
50%	4323.000000	3.000000	7.000000	12.000000	0.000000	3.000000	1.000000	
75%	6484.000000	3.000000	10.000000	18.000000	0.000000	5.000000	1.000000	
max	8645.000000	4.000000	12.000000	23.000000	1.000000	6.000000	1.000000	

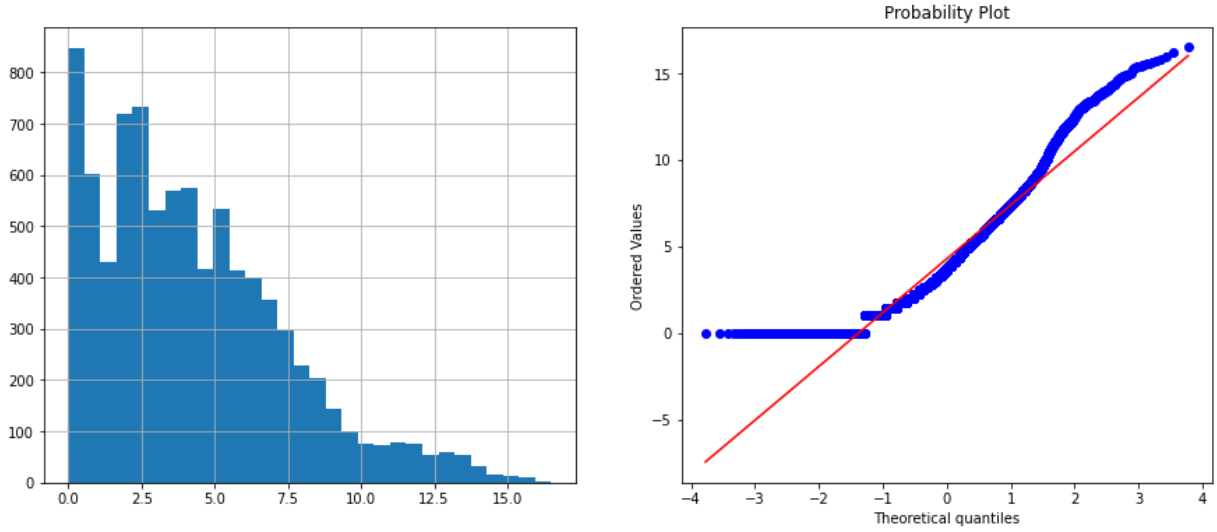
Задача №14.

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "квадратный корень"

Нормализацию будем проводить для поля casual

In [6]:
```
import matplotlib.pyplot as plt
import scipy.stats as stats
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    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()
```

In [7]:
```
data['casual_sqr'] = data['casual']**(1/2)
diagnostic_plots(data, 'casual_sqr')
```



In [8]:
```
c = pd.DataFrame({'casual':data['casual'], 'casual_sqr':data['casual_sqr']})
c
```

Out[8]:

	casual	casual_sqr
0	3	1.732051
1	8	2.828427
2	5	2.236068
3	3	1.732051
4	0	0.000000
...	...	...
8640	19	4.358899
8641	8	2.828427

	casual	casual_sqr
8642	2	1.414214
8643	2	1.414214
8644	4	2.000000

8645 rows × 2 columns

Задача №24.

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод вложений (embedded method). Используйте подход на основе линейной или логистической регрессии (в зависимости от того, на решение какой задачи ориентирован выбранный Вами набор данных - задачи регрессии или задачи классификации).

In [9]:
```
data = pd.read_csv('WineQT.csv', sep=",")
```

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [11]:
```
data_x = data.drop('quality', 1).values
data_y = data["quality"]
```

```
/tmp/ipykernel_3100244/3300780049.py:1: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be keyword-onl
y
  data_x = data.drop('quality', 1).values
```

In [12]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
```

Логистическая регрессия

In [13]:
```
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=500, r
e_lr1.fit(data_x, data_y)
# Коэффициенты регрессии
e_lr1.coef_
```

Out[13]:
```
array([[ 7.31773358e-01,  1.13568377e+01,  8.91186084e+00,
         3.04216165e-01,  4.37110143e+00, -1.09125647e-01,
        -5.12893327e-02, -9.33578279e+00,  1.04438044e+01,
        -4.41751206e+00, -3.50014778e+00,  4.80449560e-03],
       [ 3.81319659e-02,  3.86245431e+00, -1.49611893e-01,
         9.69989775e-02,  3.63053351e+00,  1.84713100e-02,
        -1.38966434e-02, -7.51419677e+00,  3.04691650e+00,
         8.81381252e-01, -2.28879254e-01, -5.70916398e-04],
       [-1.53673565e-01,  2.21015757e+00,  1.23863208e+00,
        -1.41329631e-02,  2.33258535e+00, -1.52973103e-02,
         1.46665248e-02,  5.89658686e+00, -3.72280302e-01,
        -2.94654021e+00, -9.30628787e-01,  1.66672278e-05],
       [ 1.85039500e-01, -1.88505338e+00, -2.22614493e+00,
        -5.36600797e-02,  1.82940964e+00,  1.42108720e-02,
        -9.70046454e-03, -2.96616397e+00,  8.57432047e-01,
         1.06568136e+00,  2.22194584e-01,  2.51674443e-04],
       [-1.46754625e-02, -4.08461361e+00,  8.77696845e-01,
         1.20299693e-01, -9.84922310e+00,  1.40230707e-02,
        -1.17069229e-02, -3.49981788e+00, -9.89248318e-01,
         2.51391766e+00,  8.05424748e-01, -3.28118493e-04],
       [-5.74881152e-01,  6.29897165e-01,  4.88729259e+00,
        -1.22172694e-01, -2.40502797e+01, -2.32794982e-02,
        -2.01728241e-02,  5.21520529e+00, -6.17946461e+00,
         2.75243277e+00,  1.01225399e+00, -5.62490555e-04]])
```

In [14]:
```
# Все признаки являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(data_x, data_y)
sel_e_lr1.get_support()
```

Out[14]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True])
```

Линейная регрессия

In [15]:
```
# Используем L1-регуляризацию
e_ls1 = Lasso(random_state=1)
e_ls1.fit(data_x, data_y)
# Коэффициенты регрессии
e_ls1.coef_
```

Out[15]:
```
array([ 0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00, -3.43806518e-03, -0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  9.03280797e-05])
```

In [16]:
```
sel_e_ls1 = SelectFromModel(e_ls1)
sel_e_ls1.fit(data_x, data_y)
sel_e_ls1.get_support()
```

Out[16]:
```
array([False, False, False, False, False, False,  True, False, False,
       False, False,  True])
```

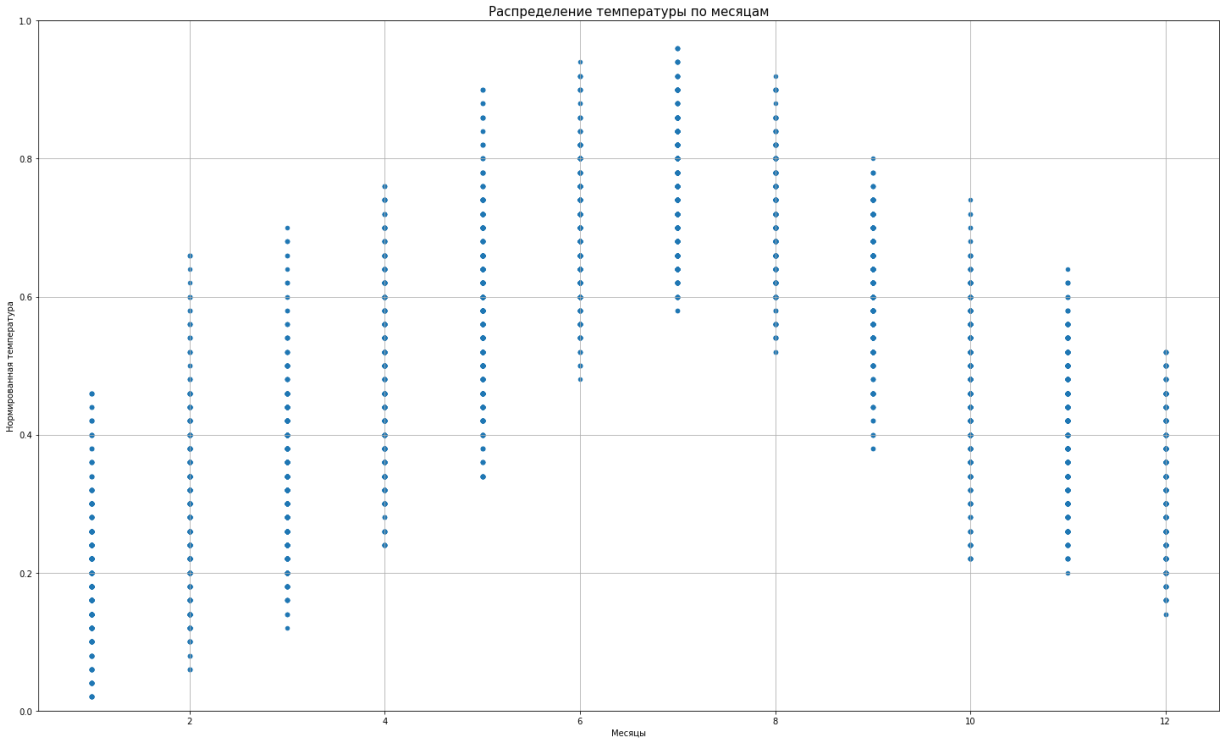
Дополнительное задание

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

Построим диаграмму рассеяния, демонстрирующую зависимость температуры от месяца года

In [17]:
```
data = pd.read_csv('bike-hour.csv', sep=",")
```

In [18]:
```
data.plot(x='mnth', y='temp', kind='scatter',  figsize=(25, 15)) ;
plt.title(f'Распределение температуры по месяцам', fontsize=15);
plt.ylim(0,1);
plt.ylabel('Нормированная температура');
plt.xlabel('Месяцы');
plt.grid(True);
```



In [ ]:
```

```