### 0.0.1. Подопригорова Н.С. ИУ5-24М

Вариант 9

**Задача №9.**

Для набора данных проведите устранение пропусков для одного (произвольного) числового признака с использованием метода заполнения “хвостом распределения”.

**Задача №29.**

Для набора данных проведите удаление константных и псевдоконстантных признаков.

Для студентов группы ИУ5-24М - для произвольной колонки данных построить график “Скрипичная диаграмма (violin plot)”.

[12]:
```
import sklearn

from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats

import matplotlib.pyplot as plt
```

[2]:
```
data = pd.read_csv('weatherAUS.csv', parse_dates=['Date'])
```

# 1. Задача 9

Выделим числовые признаки

[3]:
```
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%.
```

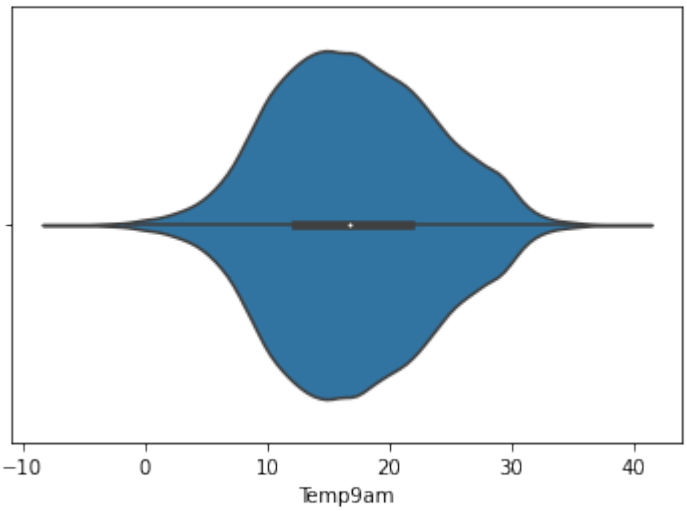
[56]:
```
num_column = 'Temp9am'
```

[57]:
```
data[num_column].isnull().sum()
```

[57]:
```
904
```

[58]:
```
sns.violinplot(x=data[num_column])
```

[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8166660640>
```

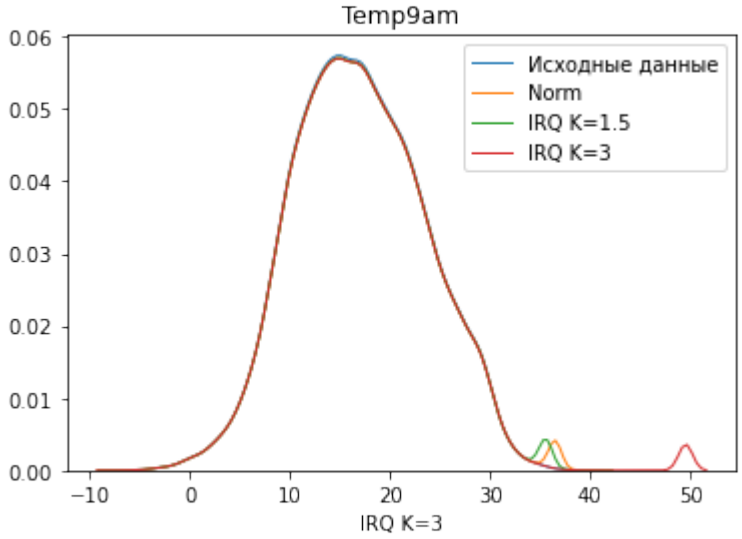



[59]:
```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """

    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
```

```
                                fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data
```

[60]:
```
fill_value1 = data[num_column].mean() + 3*data[num_column].std()
IQR = data[num_column].quantile(0.75) - data[num_column].quantile(0.25)
fill_value2 = data[num_column].quantile(0.75) + 1.5*IQR
fill_value3 = data[num_column].quantile(0.75) + 3*IQR

fill_values = [fill_value1,fill_value2,fill_value3]
strategy_params_names = ['Norm','IRQ K=1.5','IRQ K=3']

original_temp_data = data[[num_column]].values
size = original_temp_data.shape[0]
original_data = original_temp_data.reshape((size,))

new_df = pd.DataFrame({'          ':original_data})

for i in range(len(fill_values)):
    fill = fill_values[i]
    col_name = strategy_params_names[i]
    temp_data, _, _ = impute_column(data, num_column, 'constant',␣
 ↪fill_value_param=fill)
    new_df[col_name] = temp_data
```

[67]:
```
for col in new_df.columns:
    sns.distplot(new_df[col], hist = False, kde = True,
            kde_kws = {'linewidth': 1},
            label = col).set_title(num_column)
```



## 2. Задача 29

[68]:
```
from sklearn.feature_selection import VarianceThreshold
```

Создадим искусственный набор данных с константным и псевдоконстантным признаками

[76]:
```
lst_arr = [[1,2,1,21], [20,2,2,22], [15,2,3,21], [1,2,4,21],
                [1,2,5,22], [10,1,6,21], [3,2,7,21], [12,1,8,21],␣
 ↪[15,2,9,21], [17,2,10,22]]
arr = np.array(lst_arr)
data2 = pd.DataFrame(arr, columns=['f1', 'f2', 'f3', 'f4'])
data2
```

[76]:

| | f1 | f2 | f3 | f4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 21 |
| 1 | 20 | 2 | 2 | 22 |
| 2 | 15 | 2 | 3 | 21 |
| 3 | 1 | 2 | 4 | 21 |
| 4 | 1 | 2 | 5 | 22 |
| 5 | 10 | 1 | 6 | 21 |
| 6 | 3 | 2 | 7 | 21 |
| 7 | 12 | 1 | 8 | 21 |
| 8 | 15 | 2 | 9 | 21 |
| 9 | 17 | 2 | 10 | 22 |

[78]:
```
selector = VarianceThreshold(threshold=0.25)
selector.fit(data2)

selector.variances_
```

[78]:
```
array([49.25,  0.16,  8.25,  0.21])
```

[79]:
```
selector.transform(data2)
```

[79]:
```
array([[ 1,  1],
       [20,  2],
       [15,  3],
       [ 1,  4],
       [ 1,  5],
       [10,  6],
       [ 3,  7],
       [12,  8],
       [15,  9],
       [17, 10]])
```

У двух признаков (f2, f4) дисперсия меньше установленной пороговой, так что они были удалены.

## 3. Скрипичная диаграмма

[80]:
```
sns.violinplot(x=data[num_column])
```

[80]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8167c1eaf0>
```