Московский Государственный Технический Университет имени Н.Э.Баумана

Факультет Информатика и системы управления

Кафедра ИУ-5

«Системы обработки информации и управления»

Отчёт по Рубежному контролю No 1

Методы обработки данных

Выполнили стуленты группы ИУ-5

Шэнь Цюцзе 22М

Москва 2022г.

Номер вариант: 22
Номер задачи №1: 7
Номер задачи №2: 33

Дополнительные требования по группам: для произвольной колонки данных построить гистограмму.

Задача №7: Для набора данных проведите устранение пропусков для одного (произвольного) числового признака с использованием метода заполнения медианой.

Задача №33: Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), алгоритм полного перебора (exhaustive feature selection).

Задача №7

import library and data

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn.svm import SVR
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import Lasso
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import SelectKBest, SelectPercentile
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")
```

```
# Будем использовать только обучающую выборку
rdata = pd.read_csv('/content/drive/MyDrive/DataSets/All-Planets-Dataset.csv')
old_shape = rdata.shape
#rdata = rdata.dropna() # 删除表中带有空值的行
new_shape = rdata.shape
print('old shape:', old_shape, '\tnew shape:', new_shape)
```

```
old shape: (822, 9)     new shape: (822, 9)
```

+ 代码 + 文本

[6]
```
rdata
```

	name	mass	radius	period	semi_major_axis	temperature	distance_light_year	host_star_mass	host_star_temperature
0	HD 240210 b	5.2100	NaN	501.750000	1.16000	NaN	465.96	0.82	4297.0
1	Gliese 1214 b	0.0197	0.254	1.580405	0.01411	547.0	47.78	0.15	3026.0
2	CoRoT-30 b	2.9000	1.009	9.060050	0.08440	NaN	3100.00	0.98	5650.0
3	HD 4203 b	2.2300	NaN	431.880000	1.17000	NaN	266.05	1.25	5596.0
4	HD 4203 c	2.1700	NaN	6700.000000	6.95000	NaN	253.80	1.13	5702.0

```
list(zip(rdata.columns, [i for i in rdata.dtypes]))
```

```
[('name', dtype('O')),
 ('mass', dtype('float64')),
 ('radius', dtype('float64')),
 ('period', dtype('float64')),
 ('semi_major_axis', dtype('float64')),
 ('temperature', dtype('float64')),
 ('distance_light_year', dtype('float64')),
 ('host_star_mass', dtype('float64')),
 ('host_star_temperature', dtype('float64'))]
```

[8]
```
# 找出带有空值的列
hcols_with_na = [c for c in rdata.columns if rdata[c].isnull().sum() > 0]
hcols_with_na
```

```
['mass',
 'radius',
 'period',
 'semi_major_axis',
 'temperature',
 'distance_light_year',
 'host_star_mass',
 'host_star_temperature']
```

[9]
```
rdata.shape
```

```
(822, 9)
```

[10]
```
# 统计每一列的空值数量
[(c, rdata[c].isnull().sum()) for c in hcols_with_na]
```

```
[('mass', 44),
 ('radius', 627),
 ('period', 21),
 ('semi_major_axis', 42),
 ('temperature', 640),
 ('distance_light_year', 11),
 ('host_star_mass', 18),
 ('host_star_temperature', 31)]
```

```
# 统计每一列中空值所占的百分比
[(c, rdata[c].isnull().mean()) for c in hcols_with_na]
```

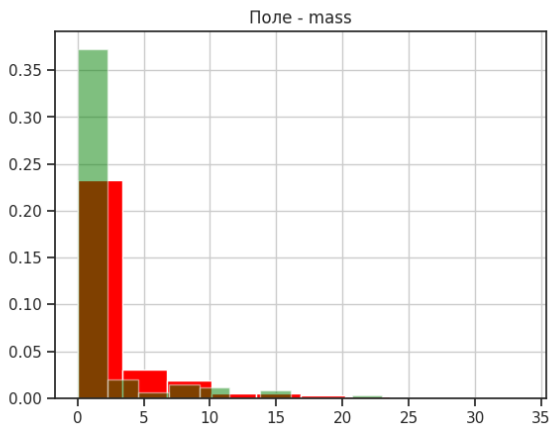
```
[('mass', 0.0535279805352798),
 ('radius', 0.7627737226277372),
 ('period', 0.025547445255474453),
 ('semi_major_axis', 0.051094890510948905),
 ('temperature', 0.7785888077858881),
 ('distance_light_year', 0.01338199513381995),
 ('host_star_mass', 0.02189781021897810 3),
 ('host_star_temperature', 0.037712895377128956)]
```

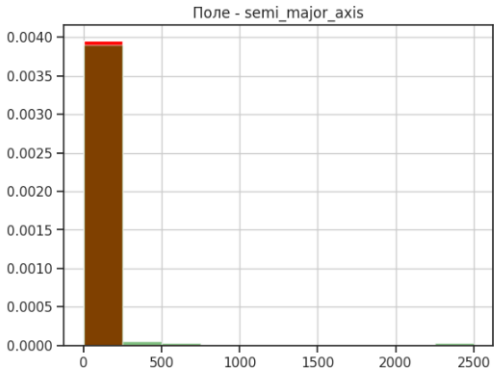
[12]
```
# 剔除缺失值较多的列，并剔除有缺失值的行
#rcols_intersted = ['name', 'mass', 'period', 'semi_major_axis', 'distance_light_year', 'host_star_mass', 'host_star_temperature']
rcols_intersted = ['mass', 'semi_major_axis', 'temperature', 'distance_light_year', 'host_star_mass', 'host_star_temperature']
rdata_drop = rdata[rcols_intersted].dropna(axis=0, how='any')
print(rdata.shape, rdata_drop.shape)
```

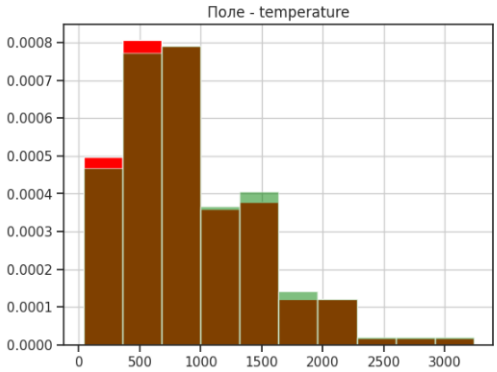
```
(822, 9) (154, 6)
```

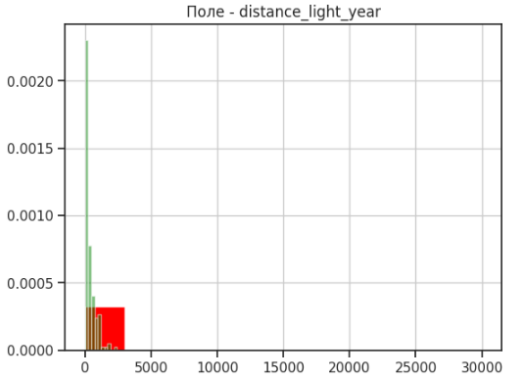
[13]
```
def plot_hist_diff(old_ds, new_ds, cols):
    """
    Разница между распределениями до и после устранения пропусков
    """
    for c in cols:
        fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.title.set_text('Поле - ' + str(c))
        old_ds[c].hist(bins=10, ax=ax, color='red', density=True)
        new_ds[c].hist(bins=10, ax=ax, color='green', density=True, alpha=0.5)
        plt.show()
```

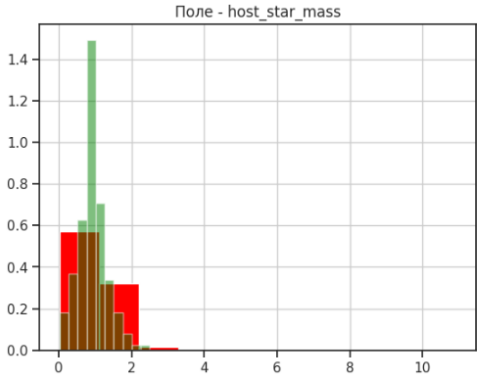
```
plot_hist_diff(rdata[rcols_intersted], rdata_drop, rcols_intersted)
```

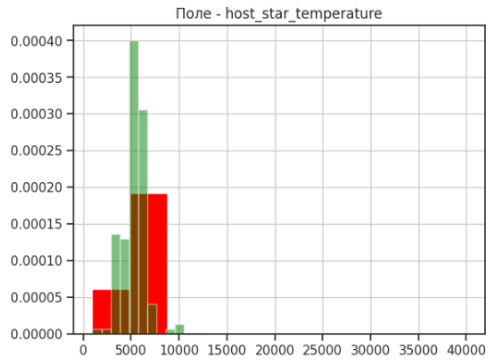



Поле - semi_major_axis
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000
2500


Поле - temperature
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
500
1000
1500
2000
2500
3000


Поле - distance_light_year
0.0020
0.0015
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
25000
30000


Поле - host_star_mass
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10



Задача №33

алгоритм полного перебора (exhaustive feature selection)

```
!pip install -U mlxtend
```

```
[28] from sklearn.datasets import make_classification
     from sklearn.neighbors import KNeighborsClassifier
     from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

     # 生成二分类数据集
     X, y = make_classification(n_samples=100, n_features=6, n_informative=3, n_redundant=0, random_state=42)

     # 定义KNN分类器
     knn = KNeighborsClassifier(n_neighbors=3)

     # 定义特征选择器
     efs = EFS(knn,
               min_features=2,
               max_features=4,
               scoring='accuracy',
               print_progress=True,
               cv=5)

     # 进行特征选择
     efs = efs.fit(X, y)

     # 输出结果
     print('Best accuracy score: %.2f' % efs.best_score_)
     print('Best subset (indices):', efs.best_idx_)
     print('Best subset (corresponding names):', [f'Feature {idx+1}' for idx in efs.best_idx_])
```

```
Features: 50/50Best accuracy score: 0.90
Best subset (indices): (0, 3)
Best subset (corresponding names): ['Feature 1', 'Feature 4']
```