**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**



**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

**Методические указания к лабораторным работам по курсу «Машинное обучение»**

**Лабораторная работа №2**

**«Обработка признаков(часть1)**

Выполнил Уляшин В.В. (ИУ5-23М)

Москва, 2023 г.

# ЗАДАНИЕ

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

* 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.
* Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  + устранение пропусков в данных;
  + кодирование категориальных признаков;
  + нормализация числовых признаков.

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
import pandas as pd  
from matplotlib import pyplot as plt  
import seaborn as sns

path="titanic.csv"  
data = pd.read\_csv(path)

data.head()

PassengerId Survived Pclass \  
0 1 0 3   
1 2 1 1   
2 3 1 3   
3 4 1 1   
4 5 0 3   
  
 Name Sex Age SibSp \  
0 Braund, Mr. Owen Harris male 22.0 1   
1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1   
2 Heikkinen, Miss. Laina female 26.0 0   
3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1   
4 Allen, Mr. William Henry male 35.0 0   
  
 Parch Ticket Fare Cabin Embarked   
0 0 A/5 21171 7.2500 NaN S   
1 0 PC 17599 71.2833 C85 C   
2 0 STON/O2. 3101282 7.9250 NaN S   
3 0 113803 53.1000 C123 S   
4 0 373450 8.0500 NaN S

data.shape

(891, 12)

data\_features = list(zip(  
 [i for i in data.columns],  
 zip(  
 [str(i) for i in data.dtypes],  
 [i for i in data.isnull().sum()])))  
data\_features

[('PassengerId', ('int64', 0)),  
 ('Survived', ('int64', 0)),  
 ('Pclass', ('int64', 0)),  
 ('Name', ('object', 0)),  
 ('Sex', ('object', 0)),  
 ('Age', ('float64', 177)),  
 ('SibSp', ('int64', 0)),  
 ('Parch', ('int64', 0)),  
 ('Ticket', ('object', 0)),  
 ('Fare', ('float64', 0)),  
 ('Cabin', ('object', 687)),  
 ('Embarked', ('object', 2))]

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

[('PassengerId', 0.0),  
 ('Survived', 0.0),  
 ('Pclass', 0.0),  
 ('Name', 0.0),  
 ('Sex', 0.0),  
 ('Age', 0.19865319865319866),  
 ('SibSp', 0.0),  
 ('Parch', 0.0),  
 ('Ticket', 0.0),  
 ('Fare', 0.0),  
 ('Cabin', 0.7710437710437711),  
 ('Embarked', 0.002244668911335578)]

# Колонки с пропусками  
hcols\_with\_na = [c for c in data.columns if data[c].isnull().sum() > 0]  
hcols\_with\_na

['Age', 'Cabin', 'Embarked']

[(c, data[c].isnull().sum()) for c in hcols\_with\_na]

[('Age', 177), ('Cabin', 687), ('Embarked', 2)]

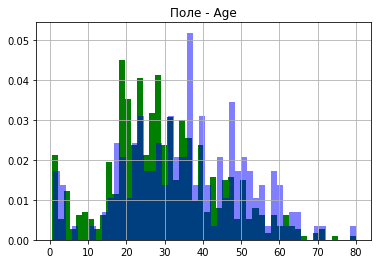
# Колонки для которых удаляются пропуски  
hcols\_with\_na\_temp = ['Age', 'Cabin', 'Embarked']

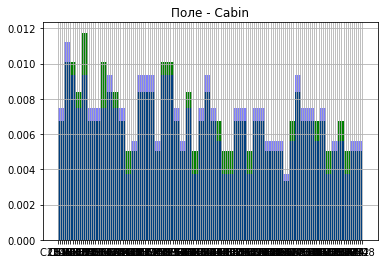
# Удаление пропусков  
data\_drop = data[hcols\_with\_na\_temp].dropna()  
data\_drop.shape

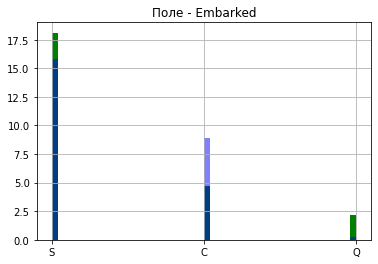
(183, 3)

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=50, ax=ax, density=True, color='green')  
 new\_ds[c].hist(bins=50, ax=ax, color='blue', density=True, alpha=0.5)  
 plt.show()

plot\_hist\_diff(data, data\_drop, hcols\_with\_na\_temp)







# Удалим колонки с высоким процентом (более 50%) пропусков:  
data\_col\_drop = data.dropna(axis=1, thresh=450)  
[(c, data[c].isnull().sum()) for c in data\_col\_drop]

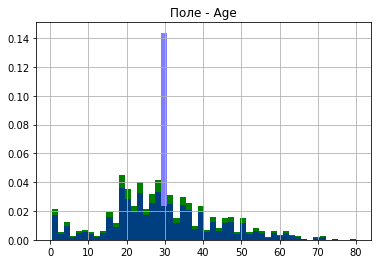
[('PassengerId', 0),  
 ('Survived', 0),  
 ('Pclass', 0),  
 ('Name', 0),  
 ('Sex', 0),  
 ('Age', 177),  
 ('SibSp', 0),  
 ('Parch', 0),  
 ('Ticket', 0),  
 ('Fare', 0),  
 ('Embarked', 2)]

# Заполним пропуски средними значениями:  
def impute\_na(df,variable,value):  
 df[variable].fillna(value,inplace=True)  
data\_fill = data\_col\_drop.copy()  
impute\_na(data\_fill,'Age',data['Age'].mean())

data\_fill.isnull().sum()

PassengerId 0  
Survived 0  
Pclass 0  
Name 0  
Sex 0  
Age 0  
SibSp 0  
Parch 0  
Ticket 0  
Fare 0  
Embarked 2  
dtype: int64

plot\_hist\_diff(data, data\_fill, ['Age'])



# Кодирование категориальных признаков  
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()  
cat\_enc\_le=le.fit\_transform(data['Age'])

data['Age'].unique()

array([22. , 38. , 26. , 35. , nan, 54. , 2. , 27. , 14. ,  
 4. , 58. , 20. , 39. , 55. , 31. , 34. , 15. , 28. ,  
 8. , 19. , 40. , 66. , 42. , 21. , 18. , 3. , 7. ,  
 49. , 29. , 65. , 28.5 , 5. , 11. , 45. , 17. , 32. ,  
 16. , 25. , 0.83, 30. , 33. , 23. , 24. , 46. , 59. ,  
 71. , 37. , 47. , 14.5 , 70.5 , 32.5 , 12. , 9. , 36.5 ,  
 51. , 55.5 , 40.5 , 44. , 1. , 61. , 56. , 50. , 36. ,  
 45.5 , 20.5 , 62. , 41. , 52. , 63. , 23.5 , 0.92, 43. ,  
 60. , 10. , 64. , 13. , 48. , 0.75, 53. , 57. , 80. ,  
 70. , 24.5 , 6. , 0.67, 30.5 , 0.42, 34.5 , 74. ])

import numpy as np  
np.unique(cat\_enc\_le)

array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,  
 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,  
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,  
 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,  
 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,  
 85, 86, 87, 88], dtype=int64)

le.inverse\_transform([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,  
 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,  
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,  
 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,  
 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,  
 85, 86, 87, 88])

array([ 0.42, 0.67, 0.75, 0.83, 0.92, 1. , 2. , 3. , 4. ,  
 5. , 6. , 7. , 8. , 9. , 10. , 11. , 12. , 13. ,  
 14. , 14.5 , 15. , 16. , 17. , 18. , 19. , 20. , 20.5 ,  
 21. , 22. , 23. , 23.5 , 24. , 24.5 , 25. , 26. , 27. ,  
 28. , 28.5 , 29. , 30. , 30.5 , 31. , 32. , 32.5 , 33. ,  
 34. , 34.5 , 35. , 36. , 36.5 , 37. , 38. , 39. , 40. ,  
 40.5 , 41. , 42. , 43. , 44. , 45. , 45.5 , 46. , 47. ,  
 48. , 49. , 50. , 51. , 52. , 53. , 54. , 55. , 55.5 ,  
 56. , 57. , 58. , 59. , 60. , 61. , 62. , 63. , 64. ,  
 65. , 66. , 70. , 70.5 , 71. , 74. , 80. , nan])

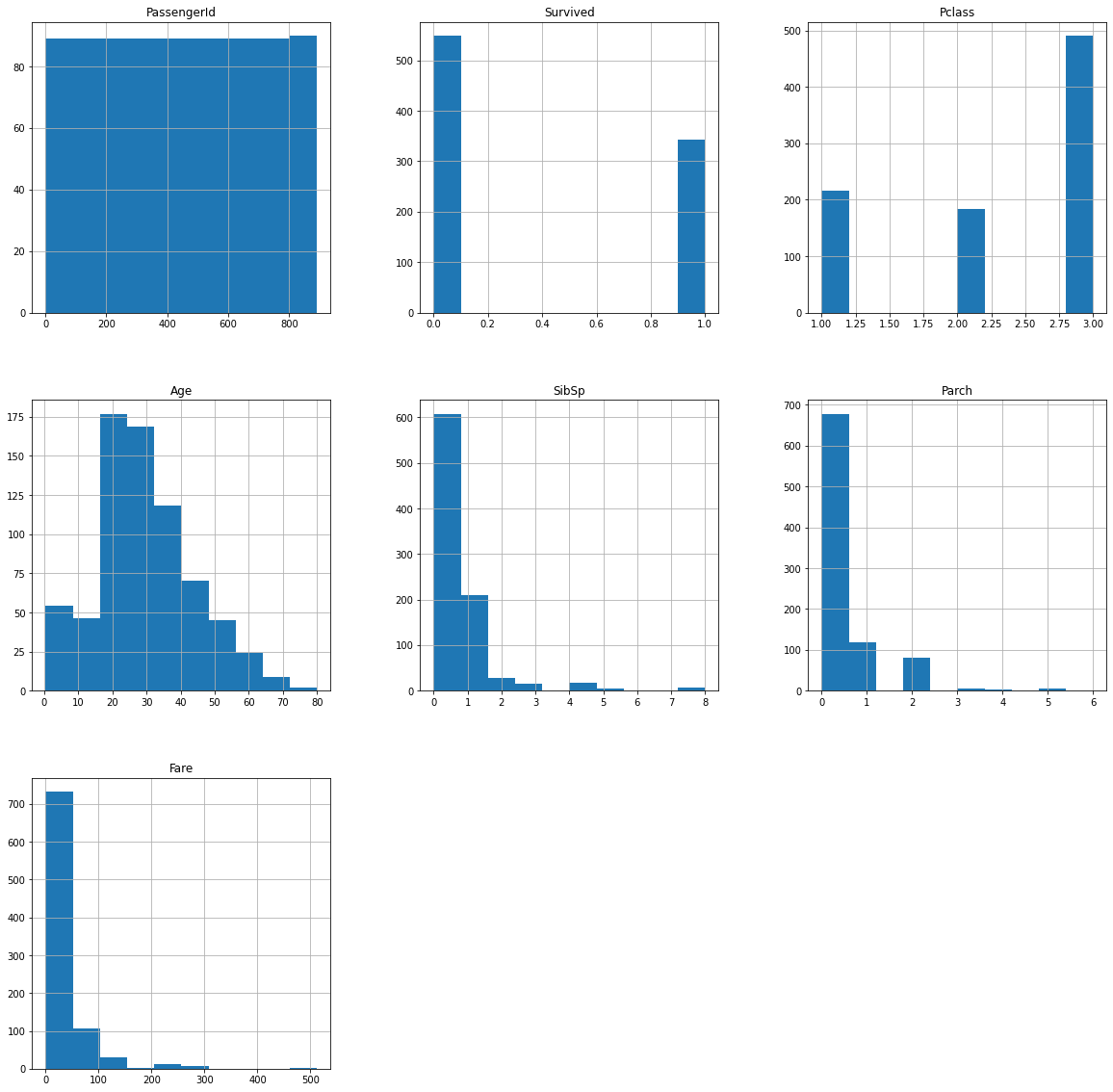
data['Pclass'].unique()

array([3, 1, 2], dtype=int64)

Нормализация цифровых признаков

from scipy.stats import stats  
def diagnostic\_plots(df,variable):  
 plt.figure(figsize=(15,6))  
 #гистрограмма  
 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()

data.hist(figsize=(20,20))  
plt.show()



normalized\_data=np.log(data\_fill['Fare'])  
  
plt.violinplot([data\_fill["Fare"],normalized\_data,])

c:\Users\stemo\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
 result = getattr(ufunc, method)(\*inputs, \*\*kwargs)  
c:\Users\stemo\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\core\function\_base.py:151: RuntimeWarning: invalid value encountered in multiply  
 y \*= step  
c:\Users\stemo\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\core\function\_base.py:161: RuntimeWarning: invalid value encountered in add  
 y += start  
c:\Users\stemo\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\lib\function\_base.py:2674: RuntimeWarning: invalid value encountered in subtract  
 X -= avg[:, None]

{'bodies': [<matplotlib.collections.PolyCollection at 0x16d23392bb0>,  
 <matplotlib.collections.PolyCollection at 0x16d2341a160>],  
 'cmaxes': <matplotlib.collections.LineCollection at 0x16d23392ca0>,  
 'cmins': <matplotlib.collections.LineCollection at 0x16d2341a820>,  
 'cbars': <matplotlib.collections.LineCollection at 0x16d2341ac10>}

