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 и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по рубежному контролю №1
Вариант №14

Выполнила:

студент группы ИУ5-53Б

Латыпова К.Н.

18.04.2021

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2020 г.

Задание:

Для заданного набора данных проведите обработку пропусков в данных для одного категориального и одного количественного признака. Какие способы обработки пропусков в данных для категориальных и количественных признаков Вы использовали? Какие признаки Вы будете использовать для дальнейшего построения моделей машинного обучения и почему?

Для студентов групп ИУ5-63Б, ИУ5Ц-83Б - для произвольной колонки данных построить график "Ящик с усами (boxplot)".

Наборы данных: Human Resources Data Set
https://www.kaggle.com/rhuebner/human-resources-data-set

Ответы на вопросы:

Для обработки пропусков данных для количественного признака Salary использовалась имьютация различными показателями центра распределения “mean”, “median”, “most_frequent” (среднее значение, медиана, мода) с помощью класса SimpleImputer библиотеки scikit-learn.

Для обработки пропусков данных для категориального признака State использовался класс SimpleImputer со стратегиями “most_frequent” или “constant” (мода и константа).

Для всего датасета можно производить удаление строк или колонок, содержащих пустые значения, или заполнение пропусков нулями. Но данные методы оказались неэффективными для данного набора данных.

Для дальнейшего построения моделей машинного обучения будут использоваться количественные признаки (перевод категориальных в количественные), т.к. многие алгоритмы машинного обучения используют количественные признаки.

Текст программы в экранные формы с примерами выполнения программы (ячейки ноутбука):

Латыпова К.Н. ИУ5-63Б

Вариант №14
Набор данных:
Human Resources Data Set

In [46]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.datasets import *
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [47]:
```
# Загрузка данных
data = pd.read_csv('HRDataset_v14.csv', sep=",")
```

In [48]:
```
# Обзор датасета
data.head()
```

Out[48]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	...	ManagerName	Mana
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	...	Michael Albert	
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	...	Simon Roup	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	...	Kissy Sullivan	

Out[48]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	...	ManagerName	Mana
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	...	Michael Albert	
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	...	Simon Roup	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	...	Kissy Sullivan	
3	Alagbe,Trina	10088	1	1	0	1	5	3	0	64991	...	Elijiah Gray	
4	Anderson, Carol	10069	0	2	0	5	5	3	0	50825	...	Webster Butler	

5 rows × 36 columns

B [49]:
```
data.shape
```

Out[49]:
```
(311, 36)
```

B [50]:
```
data.columns
```

Out[50]:
```
Index(['Employee_Name', 'EmpID', 'MarriedID', 'MaritalStatusID', 'GenderID',
       'EmpStatusID', 'DeptID', 'PerfScoreID', 'FromDiversityJobFairID',
       'Salary', 'Termd', 'PositionID', 'Position', 'State', 'Zip', 'DOB',
       'Sex', 'MaritalDesc', 'CitizenDesc', 'HispanicLatino', 'RaceDesc',
       'DateofHire', 'DateofTermination', 'TermReason', 'EmploymentStatus',
       'Department', 'ManagerName', 'ManagerID', 'RecruitmentSource',
       'PerformanceScore', 'EngagementSurvey', 'EmpSatisfaction',
       'SpecialProjectsCount', 'LastPerformanceReview_Date', 'DaysLateLast30',
       'Absences'],
      dtype='object')
```

B [51]:
```
data.dtypes
```

Out[51]:
```
Employee_Name                object
EmpID                         int64
MarriedID                     int64
MaritalStatusID               int64
GenderID                      int64
EmpStatusID                   int64
DeptID                        int64
PerfScoreID                   int64
FromDiversityJobFairID        int64
Salary                        int64
Termd                         int64
PositionID                    int64
Position                     object
State                        object
Zip                           int64
DOB                          object
Sex                          object
MaritalDesc                  object
CitizenDesc                  object
HispanicLatino               object
RaceDesc                     object
DateofHire                   object
DateofTermination            object
TermReason                   object
EmploymentStatus             object
Department                   object
ManagerName                  object
ManagerID                   float64
RecruitmentSource            object
PerformanceScore             object
EngagementSurvey            float64
EmpSatisfaction               int64
SpecialProjectsCount          int64
LastPerformanceReview_Date   object
DaysLateLast30                int64
Absences                      int64
dtype: object
```

B [52]:
```
# Проверка наличия пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

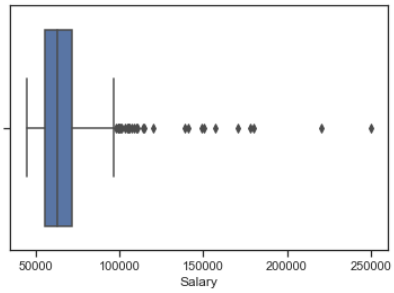
```
Employee_Name - 0
EmpID - 0
MarriedID - 0
MaritalStatusID - 0
GenderID - 0
EmpStatusID - 0
DeptID - 0
PerfScoreID - 0
FromDiversityJobFairID - 0
Salary - 0
```

```
FromDiversityJobFairID - 0
Salary - 0
Termd - 0
PositionID - 0
Position - 0
State - 0
Zip - 0
DOB - 0
Sex - 0
MaritalDesc - 0
CitizenDesc - 0
HispanicLatino - 0
RaceDesc - 0
DateofHire - 0
DateofTermination - 207
TermReason - 0
EmploymentStatus - 0
Department - 0
ManagerName - 0
ManagerID - 8
RecruitmentSource - 0
PerformanceScore - 0
EngagementSurvey - 0
EmpSatisfaction - 0
SpecialProjectsCount - 0
LastPerformanceReview_Date - 0
DaysLateLast30 - 0
Absences - 0
```

B [53]:
```
sns.boxplot(x=data['Salary'])
```

B [53]:
```
sns.boxplot(x=data['Salary'])
```

Out[53]: <AxesSubplot:xlabel='Salary'>



B [54]:
```
total_count = data.shape[0]
```

Удаление или заполнение нулями

B [55]:
```
# Удаление колонок, содержащих пустые значения
# В данном случае такое удаление колонок некорректно, так как пропуски были во всех колонках
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Удаление или заполнение нулями

B [55]:
```
# Удаление колонок, содержащих пустые значения
# В данном случае такое удаление колонок некорректно, так как пропуски были во всех колонках
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[55]: ((311, 36), (311, 34))

B [56]:
```
# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[56]: ((311, 36), (104, 36))

Второй метод тоже не показал себя эффективным, так как была удалена значительная часть строк датасета

B [57]:
```
data.head()
```

Out[57]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	...	ManagerName	Mana
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	...	Michael Albert	
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	...	Simon Roup	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	...	Kissy Sullivan	
3	Alagbe,Trina	10088	1	1	0	1	5	3	0	64991	...	Elijiah Gray	
4	Anderson, Carol	10069	0	2	0	5	5	3	0	50825	...	Webster Butler	

3	Alagbe,Trina	10088	1	1	0	1	5	3	0	64991	...	Elijah Gray
4	Anderson, Carol	10069	0	2	0	5	5	3	0	50825	...	Webster Butler

5 rows × 36 columns

B [58]:
```
# Заполнение всех пропущенных значений нулями
# Для данного датасета способ не подходит, так как содержатся категориальные признаки с пропусками
data_new_3 = data.fillna(0)
data_new_3.head()
```

Out[58]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	...	ManagerName	Mana
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	...	Michael Albert	
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	...	Simon Roup	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	...	Kissy Sullivan	
3	Alagbe,Trina	10088	1	1	0	1	5	3	0	64991	...	Elijiah Gray	
4	Anderson, Carol	10069	0	2	0	5	5	3	0	50825	...	Webster Butler	

5 rows × 36 columns

"Внедрение значений" - импьютация

Обработка пропусков для количественного признака Salary

B [59]:
```
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка ManagerID. Тип данных float64. Количество пустых значений 8, 2.57%.
```

B [60]:
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

Out[60]:

	ManagerID
0	22.0
1	4.0
2	20.0

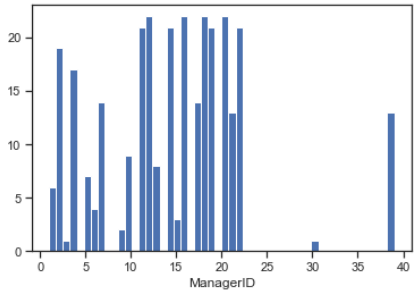
Out[60]:

	ManagerID
0	22.0
1	4.0
2	20.0
3	16.0
4	39.0
...	...
306	20.0
307	12.0
308	2.0
309	4.0
310	14.0

311 rows × 1 columns

B [61]:
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

In [61]:
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```



In [62]:
```
data_num_ManagerID = data_num[['ManagerID']]
data_num_ManagerID.head()
```

Out[62]:

	ManagerID
0	22.0
1	4.0

```
data_num_ManagerID.head()
```

Out[62]:

	ManagerID
0	22.0
1	4.0
2	20.0
3	16.0
4	39.0

In [63]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [64]:
```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_ManagerID)
mask_missing_values_only
```

Out[64]:
```
array([[False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

In [65]:
```
# Импьютация различными показателями центра распределения с помощью класса SimpleImputer
strategies=['mean', 'median', 'most_frequent']
```

In [66]:
```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_ManagerID)
    return data_num_imp[mask_missing_values_only]
```

In [67]:
```
# Среднее значение
strategies[0], test_num_impute(strategies[0])
```

Out[67]:
```
('mean',
 array([14.5709571, 14.5709571, 14.5709571, 14.5709571, 14.5709571,
        14.5709571, 14.5709571, 14.5709571]))
```

In [68]:
```
# Медиана
strategies[1], test_num_impute(strategies[1])
```

Out[68]:
```
('median', array([15., 15., 15., 15., 15., 15., 15., 15.]))
```

In [69]:
```
# Мода
strategies[2], test_num_impute(strategies[2])
```

Out[69]:
```
('most_frequent', array([12., 12., 12., 12., 12., 12., 12., 12.]))
```

Обработка пропусков для категориального признака State

In [70]:
```
# Выбор категориальных колонок с пропущенными значениями
# Цикл по колонкам датасета
```

Out[69]:
```
('most_frequent', array([12., 12., 12., 12., 12., 12., 12., 12.]))
```

Обработка пропусков для категориального признака State

В [70]:
```
# Выбор категориальных колонок с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
for col in data.columns:
    # Количество пустых значений
    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('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка DateofTermination. Тип данных object. Количество пустых значений 207, 66.56%.
```

В [71]:
```
# Импьютация с помощью класса SimpleImputer со стратегиями "most_frequent" или "constant".
cat_temp_data = data[['State']]
cat_temp_data.head()
```

Out[71]:

	State
0	MA
1	MA
2	MA

Out[71]:

	State
0	MA
1	MA
2	MA
3	MA
4	MA

В [72]:
```
cat_temp_data['State'].unique()
```

Out[72]:
```
array(['MA', 'TX', 'CT', 'VA', 'VT', 'AL', 'WA', 'CA', 'OH', 'IN', 'TN',
       'NH', 'RI', 'PA', 'CO', 'NY', 'UT', 'GA', 'FL', 'NC', 'KY', 'ID',
       'NV', 'MT', 'OR', 'ND', 'AZ', 'ME'], dtype=object)
```

В [73]:
```
cat_temp_data[cat_temp_data['State'].isnull()].shape
```

Out[73]:
```
(0, 1)
```

В [74]:
```
# Импьютация наиболее частыми значениями (мода)
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Out[74]:
```
array([['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['TX'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
```

В [75]:
```
# Пустые значения отсутствуют
np.unique(data_imp2)
```

Out[75]:
```
array(['AL', 'AZ', 'CA', 'CO', 'CT', 'FL', 'GA', 'ID', 'IN', 'KY', 'MA',
       'ME', 'MT', 'NC', 'ND', 'NH', 'NV', 'NY', 'OH', 'OR', 'PA', 'RI',
       'TN', 'TX', 'UT', 'VA', 'VT', 'WA'], dtype=object)
```

В [76]:
```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='N/A')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[76]:
```
array([['MA'],
```

Out[76]:
```
array([['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['TX'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
```

B [77]:
```
np.unique(data_imp3)
```

Out[77]:
```
array(['AL', 'AZ', 'CA', 'CO', 'CT', 'FL', 'GA', 'ID', 'IN', 'KY', 'MA',
       'ME', 'MT', 'NC', 'ND', 'NH', 'NV', 'NY', 'OH', 'OR', 'PA', 'RI',
       'TN', 'TX', 'UT', 'VA', 'VT', 'WA'], dtype=object)
```

B [78]:
```
data_imp3[data_imp3=='N/A'].size
```

Out[78]:
```
0
```


```
       ['MA'],
       ['MA'],
       ['MA'],
       ['TX'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
       ['MA'],
```

B [77]:
```
np.unique(data_imp3)
```

Out[77]:
```
array(['AL', 'AZ', 'CA', 'CO', 'CT', 'FL', 'GA', 'ID', 'IN', 'KY', 'MA',
       'ME', 'MT', 'NC', 'ND', 'NH', 'NV', 'NY', 'OH', 'OR', 'PA', 'RI',
       'TN', 'TX', 'UT', 'VA', 'VT', 'WA'], dtype=object)
```

B [78]:
```
data_imp3[data_imp3=='N/A'].size
```

Out[78]:
```
0
```


```
Для признака State я считаю более корректным внедрение константы N/A, так как название штата может и не соответствовать моде
```