

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

Лабораторная работа 3
по дисциплине
«Методы машинного обучения»
по теме
«Обработка пропусков в данных, кодирование категориальных
признаков, масштабирование данных»

ИСПОЛНИТЕЛЬ:

группа ИУ5-
23М

Чечнев А.А.
ФИО

подпись

"__" _____ 2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю. Е.
ФИО

подпись

"__" _____ 2020 г.

Москва – 2020

Лабораторная работа 3

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

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

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные 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.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
 - обработку пропусков в данных (не менее 3 признаков);
 - кодирование категориальных признаков (не менее 3 признаков);
 - масштабирование данных (не менее 3 признаков).

Задание 2.1

Задание 2.2

Датасет: Students Performance in Exams

Marks secured by the students

Content

This data set consists of the marks secured by the students in various subjects.

Acknowledgements

http://roycekimmons.com/tools/generated_data/exams

Inspiration

To understand the influence of the parents background, test preparation etc on students performance

In [2]:

```
#Start ML proj
import pandas as pd
pd.set_option('display.max.columns', 100)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(rc={'figure.figsize':(16,8)})
```

In [4]:

```
df_train = pd.read_csv('data/house_prices/train.csv')
df_test = pd.read_csv('data/house_prices/test.csv')
df = df_train
```

In [15]:

```
df.drop(columns=['Id'], inplace=True)
```

In [38]:

```
nan_counts = df.isna().sum()
nan_counts = nan_counts[nan_counts != 0]
print('All rows: %s' % df.shape[0])
print(nan_counts)
```

```
All rows: 1460
LotFrontage      259
Alley            1369
MasVnrType        8
MasVnrArea        8
BsmtQual         37
BsmtCond         37
BsmtExposure     38
BsmtFinType1     37
BsmtFinType2     38
Electrical        1
FireplaceQu     690
GarageType       81
GarageYrBlt      81
GarageFinish     81
GarageQual       81
GarageCond       81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

In [50]:

```
df_na = df[nan_cols]
df_res = df_na.copy()
```

In [51]:

```
df_na.dtypes
```

Out[51]:

```
LotFrontage    float64
Alley          object
MasVnrType     object
MasVnrArea     float64
BsmtQual       object
BsmtCond       object
BsmtExposure   object
BsmtFinType1   object
BsmtFinType2   object
Electrical     object
FireplaceQu    object
GarageType     object
GarageYrBlt    float64
GarageFinish   object
GarageQual     object
GarageCond     object
PoolQC        object
Fence         object
MiscFeature    object
dtype: object
```

In [66]:

```
for col in df_na:
    if (df_na.dtypes[col] == 'object'):
        print(col, nan_counts[col] , df_with_na[col].unique(), sep='\t')
```

```
Alley    1369    [nan 'Grv1' 'Pave']
MasVnrType    8    ['BrkFace' 'None' 'Stone' 'BrkCmn' nan]
BsmtQual    37    ['Gd' 'TA' 'Ex' nan 'Fa']
BsmtCond    37    ['TA' 'Gd' nan 'Fa' 'Po']
BsmtExposure    38    ['No' 'Gd' 'Mn' 'Av' nan]
BsmtFinType1    37    ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' nan 'LwQ']
BsmtFinType2    38    ['Unf' 'BLQ' nan 'ALQ' 'Rec' 'LwQ' 'GLQ']
Electrical    1    ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' nan]
FireplaceQu    690    [nan 'TA' 'Gd' 'Fa' 'Ex' 'Po']
GarageType    81    ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' nan 'Basment'
'2Types']
GarageFinish    81    ['RFn' 'Unf' 'Fin' nan]
GarageQual    81    ['TA' 'Fa' 'Gd' nan 'Ex' 'Po']
GarageCond    81    ['TA' 'Fa' nan 'Gd' 'Po' 'Ex']
PoolQC    1453    [nan 'Ex' 'Fa' 'Gd']
Fence    1179    [nan 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']
MiscFeature    1406    [nan 'Shed' 'Gar2' 'Othr' 'TenC']
```

In []:

In []:

In [100]:

```
all_data_na = (df_train.isnull().sum() / len(df_train)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=F
missing_data = pd.DataFrame({'Missing Ratio' :all_data_na})
missing_data
```

Out[100]:

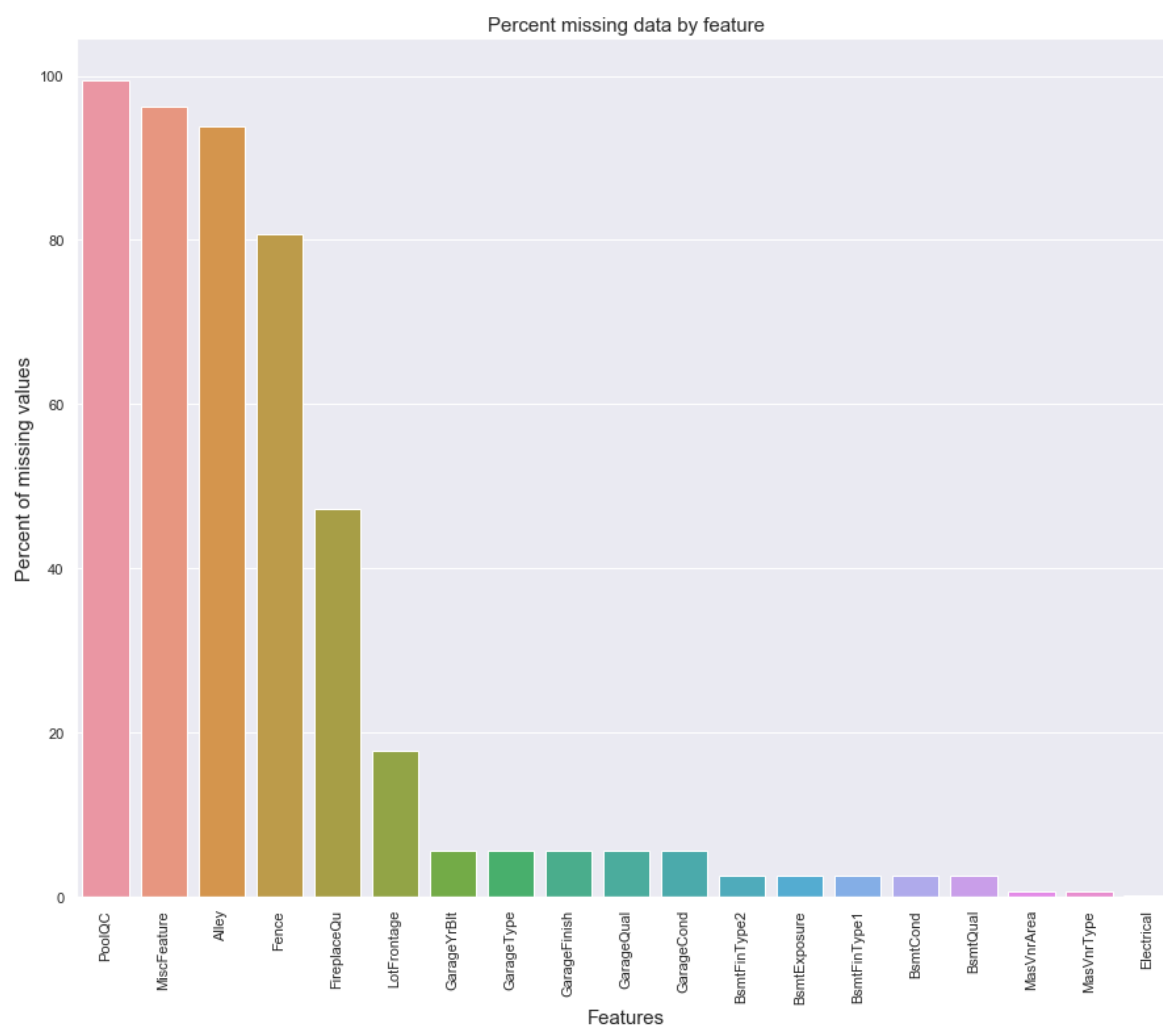
	Missing Ratio
PoolQC	99.520548
MiscFeature	96.301370
Alley	93.767123
Fence	80.753425
FireplaceQu	47.260274
LotFrontage	17.739726
GarageYrBlt	5.547945
GarageType	5.547945
GarageFinish	5.547945
GarageQual	5.547945
GarageCond	5.547945
BsmtFinType2	2.602740
BsmtExposure	2.602740
BsmtFinType1	2.534247
BsmtCond	2.534247
BsmtQual	2.534247
MasVnrArea	0.547945
MasVnrType	0.547945
Electrical	0.068493

In [101]:

```
# Percent missing data by feature
f, ax = plt.subplots(figsize=(15, 12))
plt.xticks(rotation='90')
sns.barplot(x=all_data_na.index, y=all_data_na)
plt.xlabel('Features', fontsize=15)
plt.ylabel('Percent of missing values', fontsize=15)
plt.title('Percent missing data by feature', fontsize=15)
```

Out[101]:

Text(0.5, 1.0, 'Percent missing data by feature')



Разберемся с пустыми значениями

In [104]:

```
all_data = df_train.copy()
```

In [105]:

```
col_fill_none = ['GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'GarageYrBlt', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'MasVnrType', 'FireplaceQu', 'Fence', 'Alley', 'MiscFeature', 'PoolQC', ]

for col in col_fill_none:
    all_data[col] = all_data[col].fillna('None')

all_data["LotFrontage"] = all_data.groupby("Neighborhood")["LotFrontage"].transform(lambda x: x.fillna(x.median()))
all_data["MasVnrArea"] = all_data["MasVnrArea"].fillna(0)
all_data['MSZoning'] = all_data['MSZoning'].fillna(all_data['MSZoning'].mode()[0])
all_data = all_data.drop(['Utilities'], axis=1)
all_data["Functional"] = all_data["Functional"].fillna("Typ")
all_data['Electrical'] = all_data['Electrical'].fillna(all_data['Electrical'].mode()[0])
all_data['KitchenQual'] = all_data['KitchenQual'].fillna(all_data['KitchenQual'].mode()[0])
all_data['Exterior1st'] = all_data['Exterior1st'].fillna(all_data['Exterior1st'].mode()[0])
all_data['Exterior2nd'] = all_data['Exterior2nd'].fillna(all_data['Exterior2nd'].mode()[0])
all_data['SaleType'] = all_data['SaleType'].fillna(all_data['SaleType'].mode()[0])
```

Колонка MasVnrType в пропущенных значениях содержит либо строку None, либо NaN.

Заполним пустые строки строкой 'None'

In [106]:

```
all_data_na = (all_data.isnull().sum() / len(all_data)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Missing Ratio' :all_data_na})
missing_data.head()
```

Out[106]:

Missing Ratio

In [110]:

```
df = all_data.copy()
```

In [111]:

```
df.head()
```

Out[111]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	LotCo
0	60	RL	65.0	8450	Pave	None	Reg	Lvl	In
1	20	RL	80.0	9600	Pave	None	Reg	Lvl	
2	60	RL	68.0	11250	Pave	None	IR1	Lvl	In
3	70	RL	60.0	9550	Pave	None	IR1	Lvl	Co
4	60	RL	84.0	14260	Pave	None	IR1	Lvl	

