Демьянчук Г.В. ИУ5-22М Л/Р №3

Цель лабораторной работы

Изучить способы предварительной обработки данных для дальнейшего формирования модел

Задание

Требуется:

1. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в да пунктов можно использовать несколько различных наборов данных.
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие
    - обработку пропусков в данных;
    - кодирование категориальных признаков;
    - масштабирование данных.

Ход выполнения работы

Подключим все необходимые библиотеки и настроим отображение графиков:

```
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn.impute
import sklearn.preprocessing

# Enable inline plots
%matplotlib inline

# Set plot style
sns.set(style="ticks")

# Set plots formats to save high resolution PNG
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
```

Зададим ширину текстового представления данных, чтобы в дальнейшем текст в отчёте влеза

```
pd.set_option("display.width", 70)
```

Для выполнения данной лабораторной работы возьмём набор данных по зарплатам в Огайо:

```
data = pd.read_csv("/content/ohio.csv")
```

Посмотрим на эти наборы данных:

```
data.head()
```

	Name	Job Titles	Department	Full or Part-
0	ABEJERO, JASON V	POLICE OFFICER	POLICE	
1	ABERCROMBIE IV, EARL S	PARAMEDIC I/C	FIRE	
2	ABERCROMBIE, TIMOTHY	MOTOR TRUCK DRIVER	STREETS & SAN	
3	ABFALL, RICHARD C	POLICE OFFICER	POLICE	
4	ABIOYE, ADEWOLE A	LIBRARY ASSOCIATE - HOURLY	PUBLIC LIBRARY	

```
data.dtypes
```

```
Name                 object
Job Titles           object
Department           object
Full or Part-Time    object
Salary or Hourly     object
Typical Hours       float64
Annual Salary        object
Hourly Rate          object
dtype: object
```

```
data.shape
```

```
(33161, 8)
```

Обработка пропусков в данных

Найдем все пропуски в данных:

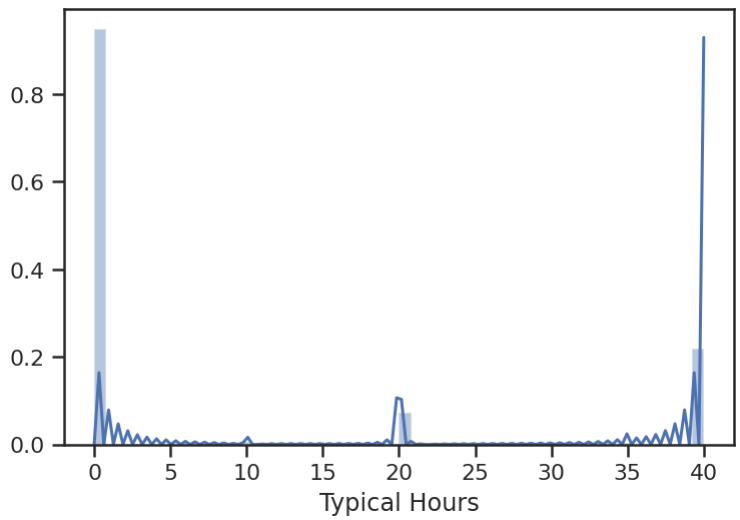
```
data.isnull().sum()
```

```
Name                     0
Job Titles               0
Department               0
Full or Part-Time        0
Salary or Hourly         0
Typical Hours        25146
Annual Salary         8015
Hourly Rate          25146
dtype: int64
```

Очевидно, что мы будем работать с колонкой `Typical Hours`.

Самый простой вариант — заполнить пропуски нулями:

```
sns.distplot(data["Typical Hours"].fillna(0));
```

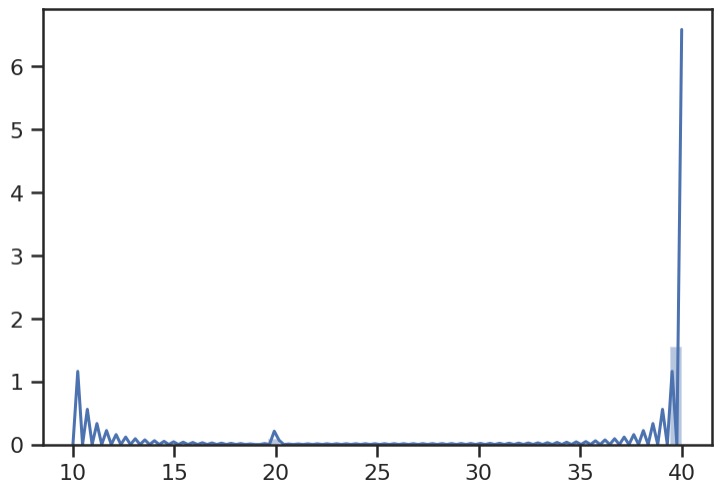


Видно, что в данной ситуации это приводит к выбросам. Логичнее было бы приложениям без часов:

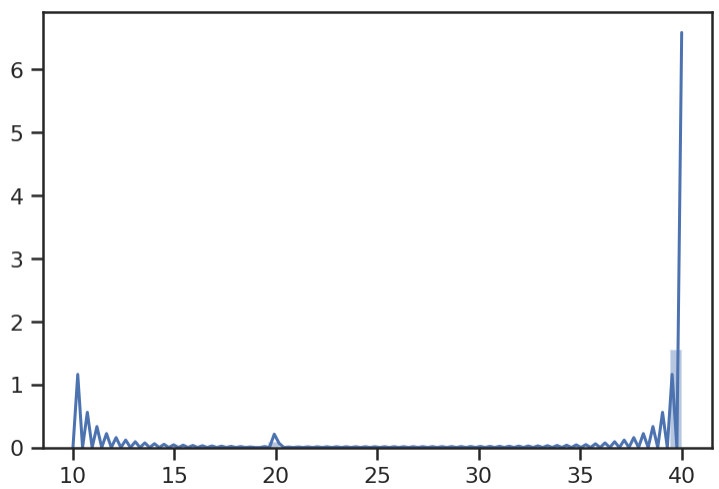
```
mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
mean_rat = mean_imp.fit_transform(data[["Typical Hours"]])
sns.distplot(mean_rat);
```

Попробуем также медианное кол-во часови самое частое кол-во часов:


```
med_imp = sklearn.impute.SimpleImputer(strategy="median")
med_rat = med_imp.fit_transform(data[["Typical Hours"]])
sns.distplot(med_rat);
```




```
freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
freq_rat = freq_imp.fit_transform(data[["Typical Hours"]])
sns.distplot(freq_rat);
```



Видно, что получили одинаковые результаты. Остановимся на обычном среднем значении:

```
data["Typical Hours"] = mean_rat
```

Кодирование категориальных признаков

Рассмотрим колонку `Salary or Hourly`:

```
types = data["Salary or Hourly"].dropna().astype(str)
types.value_counts()
```

```
Salary    25146
Hourly     8015
Name: Salary or Hourly, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

```
le = sklearn.preprocessing.LabelEncoder()
type_le = le.fit_transform(types)
print(np.unique(type_le))
le.inverse_transform(np.unique(type_le))
```

```
[0 1]
array(['Hourly', 'Salary'], dtype=object)
```

Выполним кодирование категорий наборами бинарных значений:

```
type_oh = pd.get_dummies(types)
type_oh.head()
```

	Hourly	Salary
0	0	1
1	0	1
2	1	0
3	0	1
4	1	0

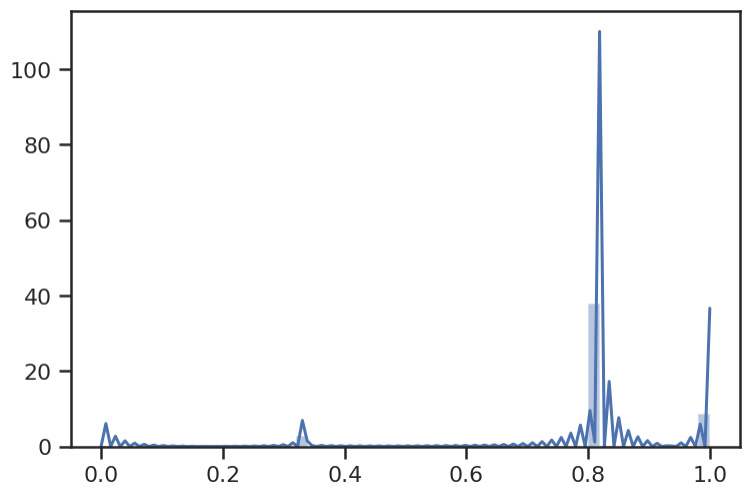
```
type_oh[type_oh["Hourly"] == 1].head()
```

	Hourly	Salary
2	1	0
4	1	0
11	1	0
14	1	0
17	1	0

Масштабирование данных

Для начала попробуем обычное MinMax-масштабирование:

```
mm = sklearn.preprocessing.MinMaxScaler()
sns.distplot(mm.fit_transform(data[["Typical Hours"]]));
```



Результат вполне ожидаемый и вполне приемлемый. Но попробуем и другие варианты, напри оценки:

```
ss = sklearn.preprocessing.StandardScaler()
sns.distplot(ss.fit_transform(data[["Typical Hours"]]));
```

Также результат ожидаемый, но его применимость зависит от дальнейшего использования.