МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ» КАФЕДРА
МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №2

з предмета “АЕФР”

Виконали: студенти 4-го курсу
Костенко Максим
Байбара Ангеліна
Калініченко Назар
Ціпаренко Ілля

Прийняла: Гуськова В.Г.

Київ 2022

1. Заповнення пропусків даних

В якості датасету було використано набір даних з попередньої роботи.

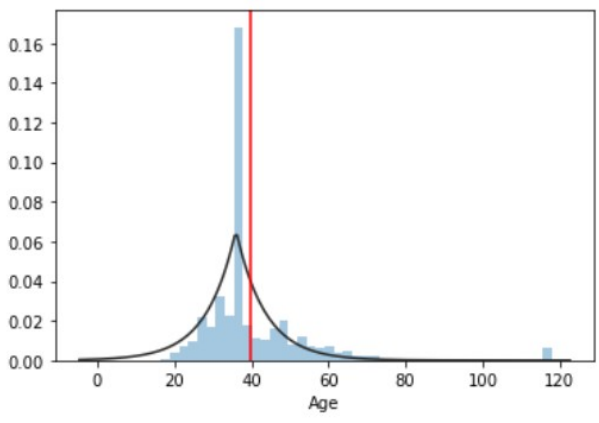
	Agency Type	Distribution Channel	Claim	Duration	NetSales	Commision (in value)	Age
0	Travel Agency	Offline	No	186	-29.0	9.57	81
1	Travel Agency	Offline	No	186	-29.0	9.57	71
2	Travel Agency	Online	No	65	-49.5	29.70	32
3	Travel Agency	Online	No	60	-39.6	23.76	32
4	Travel Agency	Online	No	79	-19.8	11.88	41

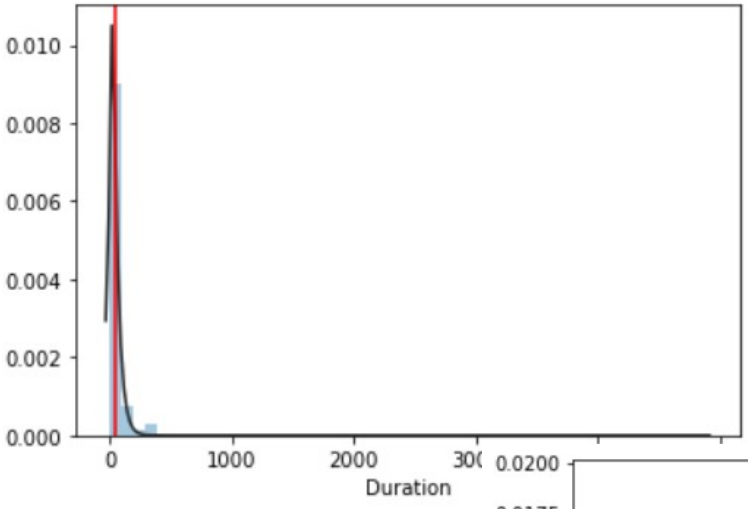
Для подальшої роботи необхідно спочатку провести певну обробку данних, а саме зробити підстановку у колонках Agency Type, Distribution Channel, Claim наступним чином:
Для Agency Type:
Travel Agency – 0;
Airlines – 1;
Для Distribution Channel:
Offline – 0;
Online – 1;
Для Claim:
No – 0;
Yes – 1;
Код та результат для цього етапу:

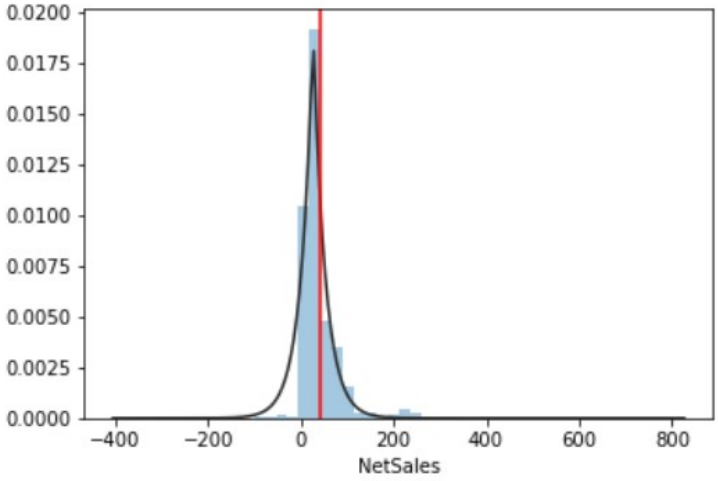
```
df["Distribution Channel"] = np.where(df["Distribution Channel"] == "Offline", 0., 1.)
df["Agency Type"] = np.where(df["Agency Type"] == "Travel Agency", 0., 1.)
df["Claim"] = np.where(df["Claim"] == "No", 0., 1.)
df['Age'] = df['Age'].astype(float)
df.head()
```

	Agency Type	Distribution Channel	Claim	Duration	NetSales	Commision (in value)	Age
0	0.0	0.0	0.0	186	-29.0	9.57	81.0
1	0.0	0.0	0.0	186	-29.0	9.57	71.0
2	0.0	1.0	0.0	65	-49.5	29.70	32.0
3	0.0	1.0	0.0	60	-39.6	23.76	32.0
4	0.0	1.0	0.0	79	-19.8	11.88	41.0

Для подальшої роботи були обрані данні із колонок Duration, NetSales та Age. Поглянемо на їх початковий розподіл та статистичні характеристики:







На графіках ми також бачимо середнє значення, відповідно для кожної колонки воно має наступне значення:
39.969980734611376 — Age;
49.31707355588542 — Duration;
40.70201797050243 — NetSales;

Після першого єтапу, додаємо у данні пропуски. Це виглядає наступним чином:

```
n_age = int(df.Age.shape[0]*0.1)
n_duration = int(df.Duration.shape[0]*0.2)
n_net_sales = int(df.NetSales.shape[0]*0.15)
    # print(n)
    # print(Data_Risks.Gender)
    # choosing random indexes to put NaN
index_nan_age = np.random.choice(df.Age.size, n_age, replace=False)
index_nan_duration = np.random.choice(df.Duration.size, n_duration, replace=False)
index_nan_net_sales = np.random.choice(df.NetSales.size, n_net_sales, replace=False)

    # adding nan to the data.
for i in index_nan_age:
    df.Age[i] = np.NaN
for i in index_nan_duration:
    df.Duration[i] = np.NaN
for i in index_nan_net_sales:
    df.NetSales[i] = np.NaN

df.head()
```

Результат роботи:

	Agency Type	Distribution Channel	Claim	Duration	NetSales	Commision (in value)	Age
0	0.0	0.0	0.0	NaN	NaN	9.57	81.0
1	0.0	0.0	0.0	NaN	-29.0	9.57	71.0
2	0.0	1.0	0.0	65.0	-49.5	29.70	NaN
3	0.0	1.0	0.0	NaN	-39.6	23.76	32.0
4	0.0	1.0	0.0	79.0	-19.8	11.88	41.0

Ми додали до наших данних пропуски із наступним співвідношенням:

```
percent_missing = df.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'percent_missing': percent_missing})
print(missing_value_df)
```

```
                     percent_missing
Agency Type                 0.000000
Distribution Channel        0.000000
Claim                       0.000000
Duration                   19.999684
NetSales                   14.998579
Commision (in value)        0.000000
Age                         9.999053
```

Тобто, для колонок Duration, NetSales та Age ми маємо відповідно 20, 15 та 10 відсодків пропущенних данних.

Для заповнення пропущенних данних використаємо алгоритми:
SimpleImputer(strategy='mean')
SimpleImputer(strategy='most-frequent')
KNNImputer(n_neighbors = 2)

Код та результат заповнення виглядае наступним чином:

```
df_imput = df.copy()
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
#test = pd.DataFrame(dataframe)
df_imput["Duration"][0] = np.NaN
df_imput.Duration = imputer.fit_transform(df_imput['Duration'].values.reshape(-1


imputer = SimpleImputer(missing_values=np.NaN, strategy='most_frequent')

df_imput.Age = imputer.fit_transform(df_imput['Age'].values.reshape(-1,1))[:,0]

# imp = IterativeImputer(max_iter=10, random_state=0)
# imp.fit(df_imput)
# df_imput = pd.DataFrame(imp.fit_transform(df_imput),columns = df.columns)

imputer = KNNImputer(n_neighbors = 2)
df_imput.NetSales = imputer.fit_transform(df_imput['NetSales'].values.reshape(-1


df_imput.head()
```

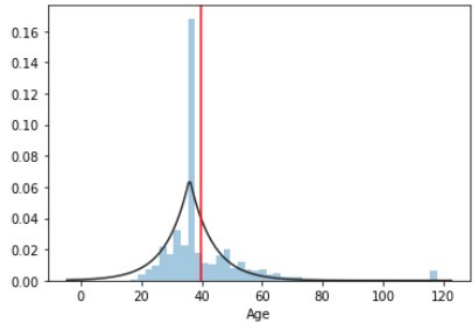
```
percent_missing = df_imput.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'percent_missing': percent_missing})
print(missing_value_df)
```

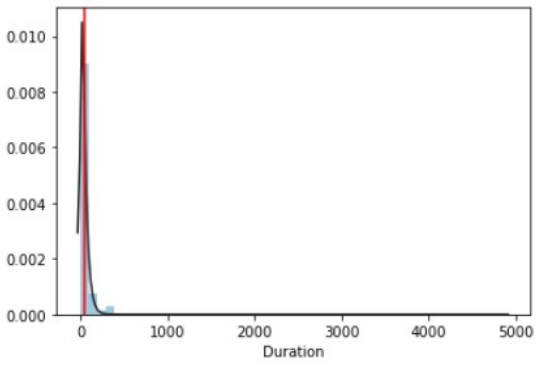
```
                       percent_missing
Agency Type                        0.0
Distribution Channel               0.0
Claim                              0.0
Duration                           0.0
NetSales                           0.0
Commision (in value)               0.0
Age                                0.0
```

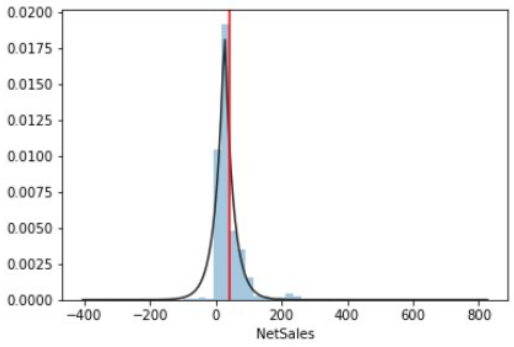
Як ми бачимо, данні були успішно заповнені. Розглянемо тепер їх вигляд та статистичні характеристики:

```
print("Means are:")
print(df_imput.Age.mean())
print(df_imput.Duration.mean())
print(df_imput.NetSales.mean())
```

```
Means are:
39.969980734611376
49.31491512041058
40.70201797050243
```







2. Фільтрація даних

Нами був обраний експоненціальний фільтр, оскільки він виявився доступний у пакеті sklearn. Для нього використовується наступна формула

$$y_k = \theta * x_k + (1-\theta) * y_{k-1}, \text{ или } y_k = y_{k-1} + \theta * (x_k - y_{k-1})$$

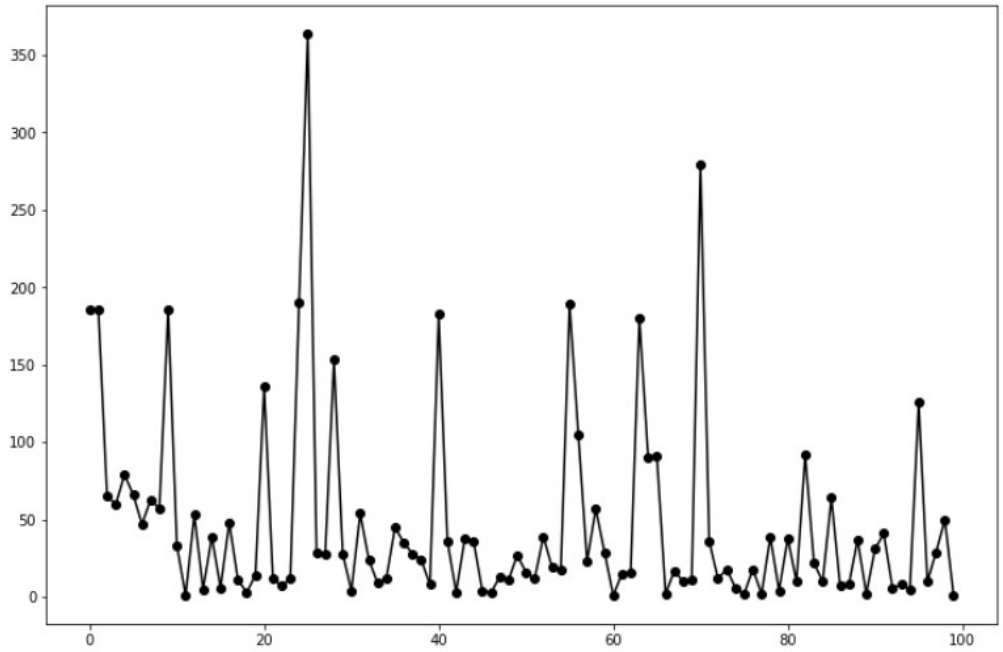
де $x_k - вимірянезначенняумоментчасук$

$y_k - фільтрованезначенняумоментчасук$

$\theta - коеффіцієнтфільтрації\,(від\,0\,до\,1)$

В якості даних для фільтрації ми узяли з тієї ж таблиці дані про тривалість поїздки — Duration.

Дані до фільтрації — були обрані перші 100 записів для наочності.



Для фільтру ми використовуємо рекомендовані у документації параметри, а саме:
Запускаємо три варіанти простого експоненційного згладжування:
1. У fit1 ми не використовуємо автоматичну оптимізацію, а замість цього вибираємо явно надати моделі параметром alpha =0.2.(синій)
У fit2, як вище, ми вибираємо alpha =0.6.(червоний)
3. У fit3 ми дозволяємо моделям статистики автоматично знайдіть для нас оптимальне значення. (зелений)

Результат: