Засоби підготовки та аналізу даних

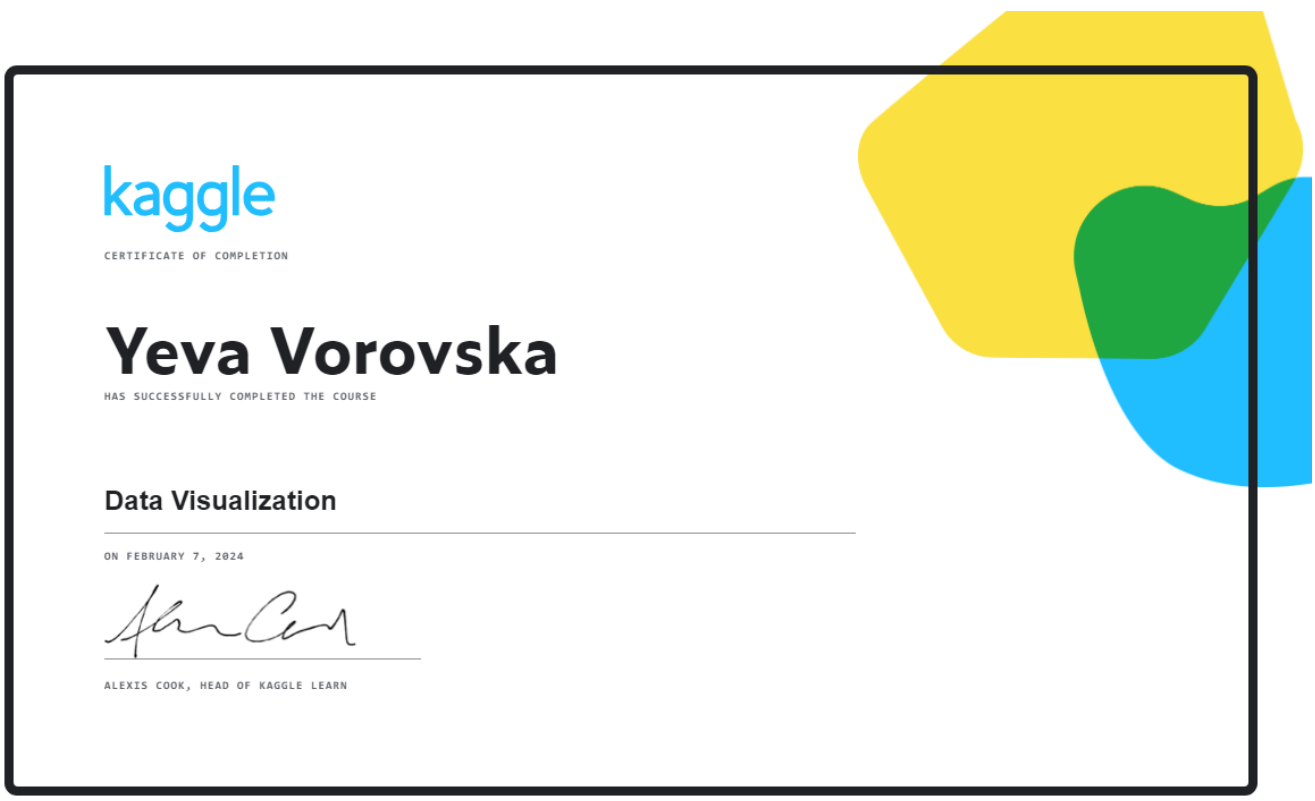
Лабораторна робота 1

ФБ-24 Воровська Єва

Посилання на репозиторій (GitHub repository):
https://github.com/yevavorov/ad_labs.git

Блок Data Visualization [Нотатки]

Сертифікат



Налаштування середовища:

```
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

Завантаження даних:

```
# Шлях до файлу
the_filepath = "../input/the.csv"

# Читання файлу
the_data = pd.read_csv(the_filepath, index_col="Column", parse_dates=True)
```

**parse_dates=True – вказує блокноту розуміти мітку кожного рядка як дату (на відміну від числа чи іншого тексту з іншим значенням).*

Вивід даних:

```
# Вивід всіх даних
the_data

# Вивід перших 5 рядків
the_data.head()

# Вивід останніх 5 рядків
the_data.tail()
```

Візуалізація даних:

```
# Визначення ширини та висоти фігури
plt.figure(figsize=(16,6))

# Додавання назви
plt.title("The Title")

# Виведення назв всіх колонок
list(the_data.columns)

# Лінійна діаграма
sns.lineplot(data=the_data)
sns.lineplot(data=the_data, label="The Label")

# Стовпчикова діаграма
sns.barplot(x=the_data.index, y=the_data['Y'])

# Теплова карта
sns.heatmap(data=the_data, annot=True)

# Точкова діаграма
sns.scatterplot(x=the_data['X'], y=the_data['Y'])
sns.scatterplot(x=the_data['X'], y=the_data['Y'], hue=the_data['Hue'])

# Лінія регресії
sns.regplot(x=the_data['X'], y=the_data['Y'])

# Декілька ліній регресії
sns.lmplot(x="X", y="Y", hue="Hue", data=the_data)

# Точкова діаграма за категоріями
```

```
sns.swarmplot(x=the_data['X'], y=the_data['Y'])

# Гістограма
sns.histplot(the_data['X'])
sns.histplot(data=the_data, x='X', hue='Hue')

# Ядрова оцінка густини розподілу (Kernel density estimation | KDE)
sns.kdeplot(data=the_data['X'], shade=True)
sns.kdeplot(data=the_data, x='X', hue='Hue', shade=True)

# 2D KDE
sns.jointplot(x=the_data['X'], y=the_data['Y'], kind="kde")

# Додавання підпису для горизонтальної осі та вертикальної осі
plt.xlabel("The X")
plt.ylabel("The Y")
```

**annot=True – значення для кожної клітинки відображатиметься на діаграмі. (Якщо це не вказано, числа будуть видалені з кожної клітинки)*

Інше:

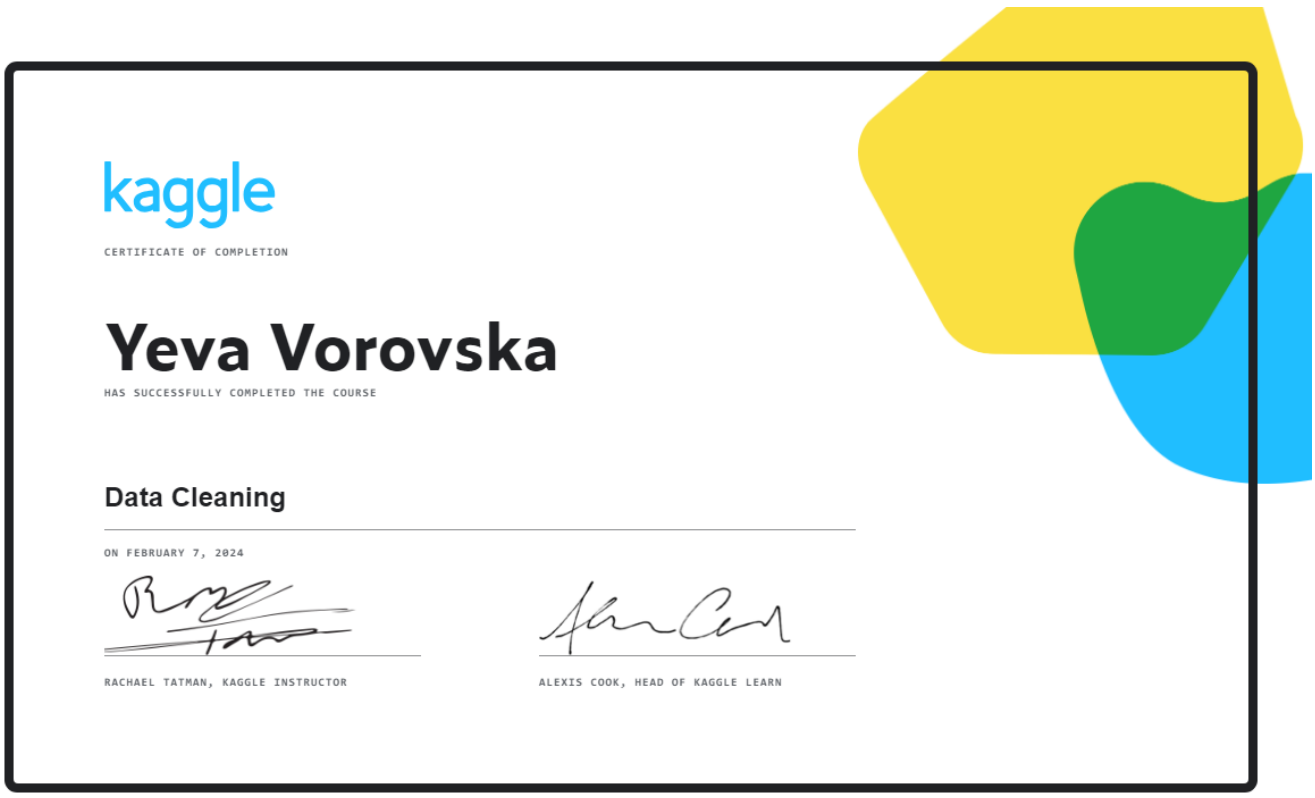
```
# Змінити стиль на "dark" тему
sns.set_style("dark")
```

**Стилі / теми:*

- *"darkgrid"*
- *"whitegrid"*
- *"dark"*
- *"white"*
- *"ticks"*

Блок Data Cleaning [Нотатки]

Сертифікат



Налаштування середовища:

```
from learntools.core import binder
binder.bind(globals())
from learntools.data_cleaning.ex1 import *
print("Setup Complete")
```

```
import pandas as pd
import numpy as np

# Читання даних
the_data = pd.read_csv("../input/the_path/the.csv")

# Визначення “сіду” (seed)
np.random.seed(0)
```

Перегляд відсутніх даних:

```
# Отримання кількості відсутніх даних на стовпець
missing_values_count = the_data.isnull().sum()

# Перегляд перших 10 відсутніх даних
missing_values_count[0:10]

# Скільки пропущених значень
```

```
total_cells = np.product(the_data.shape)
total_missing = missing_values_count.sum()

# Відсоток відсутніх даних
percent_missing = (total_missing/total_cells) * 100
print(percent_missing)
```

Прибирання відсутніх даних:

```
# Прибирання всіх рядків у яких не вистачає хоча б якихось даних
the_data.dropna()

# Прибирання всіх стовпців у яких не вистачає хоча б якихось даних
columns_with_na_dropped = the_data.dropna(axis=1)

# Скільки даних було втрачено (порівняння)
print("Columns in original dataset: %d \n" % the_data.shape[1])
print("Columns with na's dropped: %d" % columns_with_na_dropped.shape[1])
```

Заповнення відсутніх даних:

```
# Отримання невеликої підмножини набору даних
subset_the_data = the_data.loc[:, 'Column 1':'Column 2'].head()
subset_the_data

# Заміна всіх відсутніх даних на 0
subset_the_data.fillna(0)

# Заміна всіх відсутніх значень на ті, що йдуть безпосередньо після них у тому ж стов
пці, потім заміна решти відсутніх даних на 0
subset_the_data.fillna(method='bfill', axis=0).fillna(0)
```

Масштабування та нормалізація:

```
# Генерація 1000 точок даних, випадково взятих із експоненціального розподілу
original_data = np.random.exponential(size=1000)

# Масштабування даних від 0 до 1 (mix-max)
scaled_data = minmax_scaling(original_data, columns=[0])

# Побудова обох графіків для порівняння
fig, ax = plt.subplots(1, 2, figsize=(15, 3))
sns.histplot(original_data, ax=ax[0], kde=True, legend=False)
ax[0].set_title("Original Data")
sns.histplot(scaled_data, ax=ax[1], kde=True, legend=False)
ax[1].set_title("Scaled data")
plt.show()


# Нормалізація експоненціальних даних за допомогою boxcox
normalized_data = stats.boxcox(original_data)

# Побудова обох графіків для порівняння
```

```
fig, ax=plt.subplots(1, 2, figsize=(15, 3))
sns.histplot(original_data, ax=ax[0], kde=True, legend=False)
ax[0].set_title("Original Data")
sns.histplot(normalized_data[0], ax=ax[1], kde=True, legend=False)
ax[1].set_title("Normalized data")
plt.show()
```

Конвертування формату дати:

```
# Створення нового стовпця, date_parsed, із проаналізованими датами
landslides['date_parsed'] = pd.to_datetime(landslides['date'], format="%m/%d/%y")

# Отримання дня місяця зі стовпця date_parsed
day_of_month_landslides = landslides['date_parsed'].dt.day

# Прибирання відсутніх даних
day_of_month_landslides = day_of_month_landslides.dropna()

# Побудова днів місяця
sns.distplot(day_of_month_landslides, kde=False, bins=31)
```

Кодування / Декодування (Encoding / Decoding):

```
# string
before = "This is the euro symbol: €"

# Перевірка типу
type(before)
   # Out: str

# Кодування з заміною символів, що викликають помилку
after = before.encode("utf-8", errors="replace")

# Перевірка типу
type(after)
   # Out: bytes

# Перевірка як тепер виглядають байти
after
   # Out: b'This is the euro symbol: \xe2\x82\xac'

# Конвертація назад до utf-8
print(after.decode("utf-8"))
   # Out: This is the euro symbol: €
```

```
# Перегляд перших десяти тисяч байтів, щоб вгадати кодування символів
with open("../input/the_path/the.csv", 'rb') as rawdata:
    result = charset_normalizer.detect(rawdata.read(10000))

# Перевірка можливого кодування
print(result)
   # Out: {'encoding': 'utf-8', 'language': 'English', 'confidence': 1.0}
```

```
# Читання файлу враховуючи кодування вгадане charset_normalizer
the = pd.read_csv("../input/the_path/the.csv", encoding='Windows-1252')
```

Зберігання файлу з UTF-8:

```
# Зберігання файлу (буде збережено як UTF-8 за замовчуванням)
the.to_csv("the-utf8.csv")
```

Fuzzy matching:

```
# Отримання унікальних значень зі стовпця 'Country'
countries = professors['Country'].unique()

# Сортування в алфавітному порядку
countries.sort()

# Конвертація в нижній регістр
professors['Country'] = professors['Country'].str.lower()

# Видалення пробілів
professors['Country'] = professors['Country'].str.strip()

# Отримання топ 10 найбільш схожих значень до "south korea"
matches = fuzzywuzzy.process.extract("south korea", countries, limit=10, scorer=fuz
zywuzzy.fuzz.token_sort_ratio)
```

```
# Функція для заміни рядків у наданому стовпці наданого датафрейму, що відповідають н
аданому рядку вище наданого співвідношення з наданим рядком
def replace_matches_in_column(df, column, string_to_match, min_ratio = 47):
    # Отримання списку усіх унікальних значень
    strings = df[column].unique()

    # Отримання топ 10 найбільш схожих значень до наданого рядку
    matches = fuzzywuzzy.process.extract(string_to_match, strings, limit=10, scorer
=fuzzywuzzy.fuzz.token_sort_ratio)

    # Отримання значень тільки з відношенням >= 47
    close_matches = [matches[0] for matches in matches if matches[1] >= min_ratio]

    # Отримання рядків всіх близьких збігів у нашому датафреймі
    rows_with_matches = df[column].isin(close_matches)

    # Заміна всіх рядків з близькими збігами на вхідні збіги
    df.loc[rows_with_matches, column] = string_to_match


# Перевірка роботи функції
replace_matches_in_column(df=professors, column='Country', string_to_match="south k
orea")
```

Всі модулі / бібліотеки:

```
from learntools.core import binder
from learntools.data_cleaning.ex1 import *

import pandas as pd
import numpy as np

from scipy import stats

import fuzzywuzzy
from fuzzywuzzy import process

import charset_normalizer

from mlxtend.preprocessing import minmax_scaling

import seaborn as sns
import matplotlib.pyplot as plt

import datetime
```