Лабораторна робота 1
Аналіз та візуалізація даних у Python

Мета: продемонструвати свої знання про життєвий цикл аналізу даних, використовуючи заданий набір даних та вказані інструменти

Передумови / сценарій

У цій лабораторній роботі ви імпортуєте деякі пакети Python, необхідні для аналізу набору даних, що містить інформацію про злочини в Сан-Франциско. Потрібно використати засоби Python та Jupyter, щоб підготувати ці дані до аналізу, проаналізувати їх, побудувати графіки та повідомити про свої результати. Завдання до лабораторної роботи знаходиться за посиланням:

https://static-course-assets.s3.amazonaws.com/IoTFBDA201/en/course/files/2.2.4.5%20Lab%20-%20San%20Francisco%20Crime.html

Необхідні ресурси

- 1 ПК з доступом до Інтернету
- Бібліотеки Python: pandas, numpy, matplotlib, folium, datetime та csv
- Файли даних: Map-Crime_Incidents-Previous_Three_Months.csv

Частина 1: Імпорт пакетів Python

У цій частині потрібно імпортуватинаступні пакети Python, необхідні для виконання лабораторної роботи.

numpy

NumPy - це основний пакет для наукових обчислень з Python. Він містить потужний N-вимірний об'єкт масиву та складні (трансляційні) функції.

pandas

Pandas - це бібліотека з ліцензією BSD з відкритим кодом, що забезпечує високопродуктивні, прості у використанні структури даних та засоби аналізу даних для мови програмування Python.

matplotlib

Matplotlib - це бібліотека для побудови графіків для мови програмування Python та її числового математичного розширення NumPy.

Folium

Folim - це бібліотека для створення інтерактивної карти.


```
# Code cell 1
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import folium
```

Частина 2: Завантаження даних

У цій частині потрібно завантажити набір даних про злочини в Сан-Франциско (SF County) та пакети Python, необхідні для його аналізу та візуалізації.

Крок 1: Завантажте дані про злочини Сан-Франциско у дата фрейм.

На цьому кроці ви імпортуєте дані про злочини в Сан-Франциско з файлу значень, відокремлених комами (comma-separated values, CSV), у фрейм даних.

```
# code cell 2
# This should be a local path
dataset_path = './Data/Map-Crime_Incidents-Previous_Three_Months.csv'

# read the original dataset (in comma separated values format) into a
DataFrame SF = pd.read_csv(dataset_path)
```

Для перегляду перших п'яти рядків файлу csv використовується команда Linux `head`.

```
# code cell 3
!head -n 5
```

`./Data/Map-Crime_Incidents-Previous_Three_Months.csv` **Крок 2:**

Перегляньте імпортовані дані.

а) Набравши в клітинку ім'я змінної дата фрейму, ви можете структуровано візуалізувати верхні та нижні рядки.

```
# Code cell 4
pd.set_option('display.max_rows', 10) #Visualize 10 rows
SF
```

б) Використовуйте функцію `columns` для перегляду імені змінних у DataFrame.

```
# Code cell 5
SF.columns
```

Скільки змінних міститься у фреймі даних SF?

в) За допомогою функції `len` визначте кількість рядків у наборі даних.

```
# Code cell 6
len(SF)
```


Частина 3: Підготовка даних

Тепер, коли ви завантажили дані в робоче середовище, настав час підготувати дані до аналізу.

Крок 1: Витягніть місяць і день із поля Дата.

`lambda`- це ключове слово Python для визначення *анонімних функцій*. `Lambda` дозволяє вказати функцію в одному рядку коду, не використовуючи `def` не визначаючи для неї конкретного імені. Синтаксис `lambda` виразу: `lambda parameters : expression`.

Далі функція `lambda` використовується для створення вбудованої функції, яка вибирає лише цифри місяця зі змінної Date, і `int` для перетворення рядкового подання у ціле число. Потім функція *pandas* `apply` використовується для застосування цієї функції до цілого стовпця (`apply` неявно визначає цикл `for` і передає один за одним рядки до `lambda` функції). Таку саму процедуру можна зробити для дня.

```
# Code cell 7
SF['Month'] = SF['Date'].apply(lambda row: int(row[0:2]))
SF['Day'] = SF['Date'].apply(lambda row: int(row[3:5]))
```

Щоб перевірити, що ці дві змінні були додані до дата фрейму SF, використовуйте функцію `print`, щоб надрукувати деякі значення з цих стовпців і перевірити `type`, чи ці нові стовпці дійсно містять числові значення.

```
# Code cell 8
print(SF['Month'][0:2])
print(SF['Day'][0:2])

# Code cell 9
print(type(SF['Month'][0]))
```

Крок 2: Видаліть змінні з дата фрейму SF.

а) Стовпець `IncidntNum` містить багато комірок з NaN. У цьому випадку дані відсутні. Крім того, `IncidntNum`це не надає ніякого значення аналізу. Стовпець можна вилучити з дата фрейму. Одним із способів видалення небажаних змінних у фреймі даних є використання функції `del`.

```
# Code cell 10
del SF['IncidntNum']
```

б) Аналогічно, `Location` атрибут не буде в цьому аналізі. Його можна викинути з дата фрейму.

Крім того, ви можете використовувати функцію `drop` для дата фрейму, вказавши, що *вісь* дорівнює 1 (0 для рядків), і що команда не вимагає присвоєння іншому значенню для збереження результату (*inplace* = *True*).

```
# Code cell 11
```

```
SF.drop('Location', axis=1, inplace=True )
```

в) Переконайтеся, що стовпці видалено.

```
# Code cell 12
SF.columns
```


Частина 4: Аналіз даних

Тепер, коли дата фрейм підготовлений з даними, настав час проаналізувати

дані. **Крок 1: Узагальнення змінних для отримання статистичної інформації.**

а) Використовуйте функцію `value_counts` для підсумовування кількості злочинів, скоєних за типом, а потім `print` для відображення вмісту змінної *CountCategory* .

```
# Code cell 13
CountCategory = SF['Category'].value_counts()
print(CountCategory)
```

б) За замовчуванням підрахунок впорядковується за спаданням. Значення необов'язкового параметра за *зростанням* можна встановити на *True,* щоб змінити цю поведінку.

```
# Code cell 14
SF['Category'].value_counts(ascending=True)
```

Якого виду злочину було скоєно найбільше?

в) Вклавши дві функції в одну команду, ви можете досягти одного результату за допомогою одного рядка коду.

```
# Code cell 15
print(SF['Category'].value_counts(ascending=True))
```

В якому PdDistrict було найбільше випадків зареєстрованих злочинів? Надайте команди Python, які використовуються для підтримки вашої відповіді.

```
# code cell 16
# Possible code for the challenge question
print(SF['PdDistrict'].value_counts(ascending=True))
```

Крок 2. Підгрупуйте дані у менші дата фрейми (кадри даних).

а) За допомогою логічного індексування можна вибрати лише ті рядки, для яких виконується дана умова. Наприклад, наступний код витягує лише злочини, скоєні в серпні, і зберігає результат у новому DataFrame.

```
# Code cell 17
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimes
```

Скільки випадків злочинів було за серпень?
Скільки квартирних крадіжок було зареєстровано у серпні?

```
# code cell 18
# Possible code for the question: How many burglaries were reported in the month
of August?
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimesB = SF[SF['Category'] == 'BURGLARY']
len(AugustCrimesB)
```


б) Щоб створити підмножину кадру даних SF для певного дня, використовуйте операнд `query` функції для порівняння місяця та дня одночасно.

```
# Code cell 19
Crime0704 = SF.query('Month == 7 and Day == 4')
Crime0704
```

```
# Code cell 20
SF.columns
```

Частина 5: Представлення даних

Візуалізація та подання даних забезпечує миттєвий огляд, який може бути не очевидним, просто переглянувши вихідні дані. Дата фрейм SF містить координати довготи та широти, які можна використовувати для побудови даних.

Крок 1: Побудуйте графік дата фрейму SF, використовуючи змінні X та Y.
а) Використовуйте функцію `plot()`для побудови кадру даних SF. Використовуйте необов'язковий параметр, щоб побудувати графік червоним кольором і встановити фігуру маркера в коло за допомогою *ro* .

```
# Code cell 21
plt.plot(SF['X'],SF['Y'], 'ro')
plt.show()
```

б) Визначте номери відділів поліції, складіть словник *pd_districts,* щоб зв'язати їх рядок із цілим числом.

```
# Code cell 22
pd_districts = np.unique(SF['PdDistrict'])
pd_districts_levels = dict(zip(pd_districts,
range(len(pd_districts)))) pd_districts_levels
```

в) Використовуйте `apply` та, `lambda`, щоб додати ціле число поліцейського відділу до нового стовпця DataFrame

```
# Code cell 23
SF['PdDistrictCode'] = SF['PdDistrict'].apply(lambda row: pd_districts_levels[row])
```

г) Використовуйте щойно створений *PdDistrictCode* для автоматичної зміни кольору

```
# Code cell 24
plt.scatter(SF['X'], SF['Y'], c=SF['PdDistrictCode'])
plt.show()
```

Крок 2: Додайте пакети для побудови карт, щоб покращити сюжет.
На кроці 1 ви створили простий сюжет, який показує, де мали місце злочини у окрузі SF. Цей графік корисний, але `folium` надає додаткові функції, які дозволять вам накласти цей графік на карту OpenStreet.
а) `Folium` вимагає вказувати колір маркера з використанням шістнадцяткового значення. З цієї причини ми використовуємо пакет *colors* і вибираємо необхідні кольори.


```
# Code cell 25
from matplotlib import colors
districts = np.unique(SF['PdDistrict'])
print(list(colors.cnames.values())[0:len(districts)])
```

б) Створіть словник кольорів для кожного окружного відділу поліції.

```
# Code cell 26
color_dict = dict(zip(districts, list(colors.cnames.values())[0:-1:len(districts)]
))
color_dict
```

в) Створіть карту, використовуючи середні координати даних SF, щоб відцентрувати карту (за допомогою `mean`). Щоб зменшити час обчислення, використовується *plotEvery* для обмеження кількості побудованих даних. Встановіть це значення в 1 для побудови всіх рядків (візуалізація карти може зайняти багато часу).

```
# Code cell 27
# Create map
map_osm = folium.Map(location=[SF['Y'].mean(), SF['X'].mean()], zoom_start = 12)
plotEvery = 50
obs = list(zip( SF['Y'], SF['X'], SF['PdDistrict']))

for el in obs[0:-1:plotEvery]:

 folium.CircleMarker(el[0:2], color=color_dict[el[2]], fill_color=el[2],radius=
10).add_to(map_osm)

# Code cell 28
map_osm
```

Кореляційний аналіз у Python

Мета: продемонструвати свої навички кореляційного аналізу даних, використовуючи заданий набір даних та вказані інструменти

Передумови / сценарій

У цій лабораторній роботі ви дізнаєтесь, як використовувати Python для обчислення кореляції. У частині 1 ви налаштуєте набір даних. У частині 2 ви дізнаєтеся, як визначити, чи змінні в даному наборі даних є корельованими. У частині 3 ви будете використовувати Python для обчислення кореляції між двома наборами змінних. Нарешті, у частині 4 ви здійсните візуалізацію результатів дослідження.

Завдання до лабораторної роботи знаходиться за посиланням:

https://static-course-assets.s3.amazonaws.com/IoTFBDA201/en/course/files/3.1.5.5%20Lab%20-%20Correlation%20Analysis%20in%20Python.html

Необхідні ресурси

- 1 ПК з доступом до Інтернету
- Бібліотеки Python: pandas, numpy, matplotlib, seaborn
- Файли даних: brainsize.txt

Частина 1: Набір даних

Використайте набір даних, який містить зразок 40 студентів у Southwestern university. Було взято чотири субтести (словниковий запас, схожість, дизайн блоків та доповнення картинок) переглянутої Wechsler Adult Intelligence Scale (1981).

https://en.wikipedia.org/wiki/Wechsler_Adult_Intelligence_Scale

Дослідники використовували магнітно-резонансну томографію (МРТ) для визначення розміру мозку досліджуваних. Також включена інформація про стать та розмір тіла (зріст та вага). З міркувань конфіденційності дослідники утримували ваги двох предметів та зростання одного предмета. До набору даних застосовано дві прості модифікації:

1. Заміна знаків питання, що використовуються для представлення утриманих точок даних, описаних вище, рядком 'NaN'. Заміна була здійснена, оскільки Pandas неправильно обробляє знаки питання.
2. Заміна усіх символів табуляції комами, перетворивши набір даних у набір даних CSV.

Підготовлений набір даних зберігається як **brainsize.txt.**

Крок 1: Завантаження набору даних із файлу.

Перш ніж набір даних можна використовувати, його потрібно завантажити в пам’ять. У наведеному нижче коді перший рядок імпортує модулі pandas та визначається pd як дескриптор, який посилається на модуль.

Другий рядок завантажує CSV-файл набору даних у змінну з назвою brainFile. Третій рядок read_csv(), це метод pandas для перетворення CSV набору даних, збережений в brainFile в dataframe. Потім фрейм даних зберігається у змінній brainFrame.

Запустіть клітинку коду нижче, щоб виконати описані функції.

```
# Code cell 1
import pandas as pd
brainFile = './Data/brainsize.txt'
brainFrame = pd.read_csv(brainFile)
```

Крок 2: Перевірка дата фрейму.

Щоб переконатися, що фрейм даних правильно завантажений і створений, скористайтеся методом head(). Метод pandas head() відображає перші п’ять записів кадру даних.


```
# Code cell 2
brainFrame.head()
```

Частина 2: Діаграми розсіювання та корельовані змінні

Крок 1: Метод describe().

Модуль pandas включає метод describe(), який виконує однакові загальні обчислення щодо даного набору даних. На додаток до загальних результатів, включаючи кількість, середнє, стандартне відхилення, мінімальне та максимальне, describe() також є чудовим способом швидкого тестування достовірності значень у фреймі даних.

Виконайте код нижче, щоб виводити результати методу describe(), обчислені по відношенню до дата фрейму brainFrame.

```
# Code cell 3
brainFrame.describe()
```

Крок 2: Графіки діаграм розсіювання

Графіки діаграм розсіювання важливі при роботі з кореляціями, оскільки вони дозволяють швидко візуально перевірити природу взаємозв'язку між змінними.

У цій лабораторній роботі використовується коефіцієнт кореляції Пірсона, який чутливий лише до лінійного співвідношення між двома змінними (навести формули для обчислення кореляції Пірсона). Існують інші більш надійні методи кореляції, але в цій лабораторній роботі вони не розглядаються.

a. Завантажте необхідні модулі.

Перш ніж побудувати графіки, необхідно імпортувати кілька модулів, а саме numpy та matplotlib.

Запустіть код нижче, щоб завантажити ці модулі.

```
# Code cell 4
import numpy as np
import matplotlib.pyplot as plt
```

b. Відокремте дані.

Щоб результати не спотворювались через різницю в чоловічому та жіночому тілах, дата фрейм розділений на два кадри даних: один, що містить усі записи чоловіків, а інший - лише жіночі випадки. Запуск коду нижче створює два нових кадри даних, menDf і womenDf, кожен з яких містить відповідні записи.

```
# Code cell 5
menDf = brainFrame[(brainFrame.Gender == 'Male')]
womenDf = brainFrame[(brainFrame.Gender == 'Female')]
```


c. Побудуйте графіки.

Оскільки набір даних включає три різні показники інтелекту (PIQ, FSIQ та VIQ), перший рядок коду нижче використовує метод mean() для обчислення середнього значення між трьома та збереження результату у змінній menMeanSmarts. Зверніть увагу, що перший рядок також стосується menDf, відфільтрованого кадру даних, що містить лише чоловічі записи.

Другий рядок використовує matplotlib метод scatter() для створення діаграми розсіювання між menMeanSmarts змінною та MRI_Count атрибутом. MRI_Count у цьому наборі даних можна вважати мірою фізичного розміру мозку суб'єктів.

Третій рядок відображає графік.

Четвертий рядок використовується для відображення графіку.

```
# Code cell 6
menMeanSmarts = menDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(menMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```

Наведений нижче код створює графік розбіжності для відфільтрованого кадру даних лише для жінок.

```
# Code cell 7
# Graph the women-only filtered dataframe
#womenMeanSmarts = ?
#plt.scatter(?, ?)

plt.show()
%matplotlib inline
```

Частина 3: Обчислення кореляції в Python

Крок 1: Обчисліть співвідношення між brainFrame.

Метод pandas corr() забезпечує простий спосіб обчислення кореляції щодо кадру даних. Викликавши метод для фрейму даних, можна отримати кореляцію між усіма змінними одночасно.

```
# Code cell 8
brainFrame.corr(method='pearson')
```

Зверніть увагу на діагональ зліва направо у таблиці кореляцій, сформованій вище. Чому діагональ заповнена 1s? Це випадковість? Поясніть.

Не дивлячись на таблицю кореляцій вище, зауважте, що значення відображаються дзеркально; значення нижче 1 діагоналі мають дзеркальний аналог вище 1 діагоналі. Це випадковість? Поясніть. Використовуючи метод corr(), можна розрахувати кореляцію змінних, що містяться в кадрі даних лише для жінок:

```
# Code cell 9
```



```
womenDf.corr(method='pearson')
```

І те саме можна зробити для кадру даних, призначеного лише для чоловіків:

```
# Code cell 10
# Use corr() for the male-only dataframe with the pearson
method #?.corr(?)
```

Частина 4: Візуалізація

Крок 1: Встановіть Seaborn.

Для спрощення візуалізації кореляцій даних можна використовувати графіки теплових карт. На основі кольорових квадратів графіки теплових карт можуть допомогти швидко виявити кореляційні зв’язки. За допомогою модуля Python seaborn дуже легко побудувати графіки теплових карт. Спочатку запустіть код нижче, щоб завантажити та встановити модуль seaborn.

```
# Code cell 11
!pip install seaborn
```

Крок 2: Побудуйте графік кореляційної теплової карти.

Тепер, коли кадри даних готові, можна побудувати теплові карти.

Рядок 1: Створює таблицю кореляції на основі women NoGenderDf кадру даних та зберігає її в wcorr.

Рядок 2: Використовує seaborn метод heatmap() для формування та побудови графіку теплової карти. Зверніть увагу', що heatmap() приймає wcorr в якості параметра.

Рядок 3: Використовуйте для експорту та збереження сформованої теплової карти як зображення PNG. Поки рядок 3 не активний (перед ним є #символ коментаря, що змушує інтерпретатор ігнорувати його), він зберігався з інформаційною метою.

```
# Code cell 12
import seaborn as sns

wcorr = womenDf.corr()
sns.heatmap(wcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```


Подібним чином, наведений нижче код створює і складає теплову карту для кадру даних лише для чоловіків.

```
# Code cell 14
mcorr = menDf.corr()
sns.heatmap(mcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

Багато пар змінних мають кореляцію, близьку до нуля. Що це означає?

Навіщо розділяти стать?

Які змінні мають сильнішу кореляцію з розміром мозку (MRI_Count)? Це очікується? Поясніть.

Побудова лінійної регресії в Python

Мета: ознайомитись з поняттями простої лінійної регресії та роботи з наданими даними для прогнозування в Python.

Передумови / сценарій

У статистиці лінійна регресія - це спосіб моделювання взаємозв'язку між залежною змінною p і незалежною змінною x. У цій лабораторній роботі ви проаналізуєте дані про продажі району та побудуєте просту лінійну регресію для прогнозування річного чистого обсягу продажів на основі кількості магазинів у районі. Завдання до лабораторної роботи знаходиться за посиланням:

https://static-course-assets.s3.amazonaws.com/IoTFBDA201/en/course/files/4.1.2.4%20Lab%20-%20Simple%20Linear%20Regression%20in%20Python.html

Необхідні ресурси

- 1 ПК з доступом до Інтернету
- Бібліотеки Python: `pandas`, `numpy`, `scipy`, і `matplotlib`
- Файли даних: store-dist.csv

Частина 1: Імпорт бібліотек та даних

У цій частині ви імпортуєте бібліотеки та дані з файлу `stores-dist.csv`.

Крок 1: Імпортуйте бібліотеки.

На цьому кроці ви імпортуєте такі бібліотеки:

- `matplotlib.pyplot` як plt
- `numpy` як np
- `pandas` як pd

```
# Code Cell 1
```

Крок 2: Імпортуйте дані.

На цьому кроці ви імпортуєте дані з файлу `stores-dist.csv`та переконаєтесь, що файл імпортовано правильно.

```
Code Cell 2
# Import the file, stores-dist.csv
salesDist = pd.read_csv('./Data/stores-dist.csv')

# Verify the imported data
```

```
salesDist.head()
```
Заголовки стовпців `annual net sales`і `number of stores in district` перейменовані для полегшення обробки даних.

- `annual net sales` до продажів
- `number of stores in district` до магазинів

```
# Code Cell 3
# The district column has no relevance at this time, so it can be dropped.
salesDist = salesDist.rename(columns={'annual net sales':'sales','number of stores in district':'stores'})
salesDist.head()
```

Частина 2: Складання даних

Крок 1: Визначте співвідношення.

На цьому кроці ви дослідите кореляцію даних до регресійного аналізу. Ви також скинете будь-які не пов'язані між собою стовпці за необхідності.

```
# Code Cell 4
# Check correlation of data prior to doing the analysis
# # Hint: check lab 3.1.5.5
```

З коефіцієнта кореляції виявляється, що стовпець `district` має низьку кореляцію до `annual net sales`і `number of stores in the district`. Отже, колонна округа не є необхідною як частина регресійного аналізу. Колонка `District` може бути виключена з dataframe.

```
# Code Cell 5
# The district column has no relevance at this time, so it can be dropped. #sales = salesDist.drop(...)

sales.head()
```

За даними коефіцієнта кореляції, який тип кореляції ви спостерігали між річними чистими продажами та кількістю магазинів у районі?

 Введіть тут свою відповідь.

Крок 2: Створіть сюжет.

На цьому кроці ви створите графік для візуалізації даних. Ви також призначите магазини як незалежну змінну ***x*** а продажі як залежна змінна ***p***.

```
# Code Cell 6
# dependent variable for y axis
y = sales['sales']
# independent variable for x axis
x = sales.stores
```

```
# Code Cell 7
# Display the plot inline
%matplotlib inline

# Increase the size of the plot
plt.figure(figsize=(20,10))

# Create a scatter plot: Number of stores in the District vs. Annual Net
Sales plt.plot(x,y, 'o', markersize = 15)

# Add axis labels and increase the font size
plt.ylabel('Annual Net Sales', fontsize = 30)
plt.xlabel('Number of Stores in the District', fontsize = 30)

# Increase the font size on the ticks on the x and y axis
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)

# Display the scatter plot
plt.show()
```

Частина 3: Побудуйте просту лінійну регресію

У цій частині ви будете використовувати numpy для створення лінії регресії для аналізованих даних. Ви також розрахуєте центроїд для цього набору даних. Центрод - це середнє значення для набору даних. Сформована проста лінійна лінійна регресія також повинна проходити через центроїд.

Крок 1: Обчисліть нахил та перетин Y-лінії лінійної регресії.

```
# Code Cell 8
# Use numpy polyfit for linear regression to fit the data
# Generate the slope of the line (m)
# Generate the y-intercept (b)
m, b = np.polyfit(x,y,1)
print ('The slope of line is {:.2f}.'.format(m))
print ('The y-intercept is {:.2f}.'.format(b))
print ('The best fit simple linear regression line is {:.2f}x +
{:.2f}.'.format(m, b))
```


Крок 2: Обчисліть центроїд.

Центроїд набору даних обчислюється за допомогою функції середнього значення.

```
# Code Cell 9
# y coordinate for centroid
y_mean = y.mean()
# x coordinate for centroid
```

```
x_mean = x.mean()
print ('The centroid for this dataset is x = {:.2f} and y =
{:.2f}.'.format(x_mean , y_mean))
```

Крок 3: Накладіть лінію регресії та центральну точку на графіку.

```
# Code Cell 10
# Create the plot inline
%matplotlib inline

# Enlarge the plot size
plt.figure(figsize=(20,10))

# Plot the scatter plot of the data set
plt.plot(x,y, 'o', markersize = 14, label = "Annual Net Sales")

# Plot the centroid point
plt.plot(x_mean,y_mean, '*', markersize = 30, color = "r")

# Plot the linear regression line
plt.plot(x, m*x + b, '-', label = 'Simple Linear Regression Line', linewidth = 4)

# Create the x and y axis labels
plt.ylabel('Annual Net Sales', fontsize = 30)
plt.xlabel('Number of Stores in District', fontsize = 30)

# Enlarge x and y tick marks
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)

# Point out the centroid point in the plot
plt.annotate('Centroid', xy=(x_mean-0.1, y_mean-5), xytext=(x_mean-3,
y_mean-20), arrowprops=dict(facecolor='black', shrink=0.05), fontsize = 30)

# Create legend
plt.legend(loc = 'upper right', fontsize = 20)
```


Крок 4: Прогнозування

Використовуючи лінійну лінійну регресію, ви можете передбачити річний чистий обсяг продажів на основі кількості магазинів у районі.

```
# Function to predict the net sales from the regression line
def predict(query):
 if query >= 1:
```

```
 predict = m * query + b
 return predict
 else:
 print ("You must have at least 1 store in the district to predict the annu al net
sales.")

 # Code Cell 12
# Enter the number of stores in the function to generate the net sales prediction.
```

Який прогнозований чистий продаж, якщо в районі є 4 магазини?
Part 1: *Введіть тут свою відповідь.*

