

## Rapport – Analyse de données des bibliothèques Python

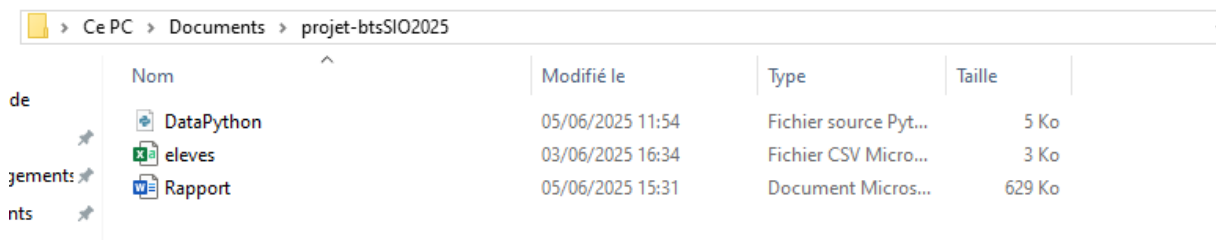
Dans le cadre de mon stage au sein de l'école polytechnique, j'ai mené un projet d'analyse de donnée d'une bibliothèque Python via la liste fournie par l'établissement, portant sur les élèves de BTS SIO. L'objectif était de déterminer si la distance entre le domicile et le lycée pouvait influencer sa moyenne générale.

### Les démarches à suivre sont les suivantes :

- Lectures des données
- Analyse des données
- Détermination d'une loi simple
- Prédiction à partir de cette loi
- Analyse des résultats
- Rédaction d'un compte rendu

### – Structuration du Projet (**BONUS**) :

Dossier « projet-btsSIO2025 » dans lequel se trouve les fichiers :



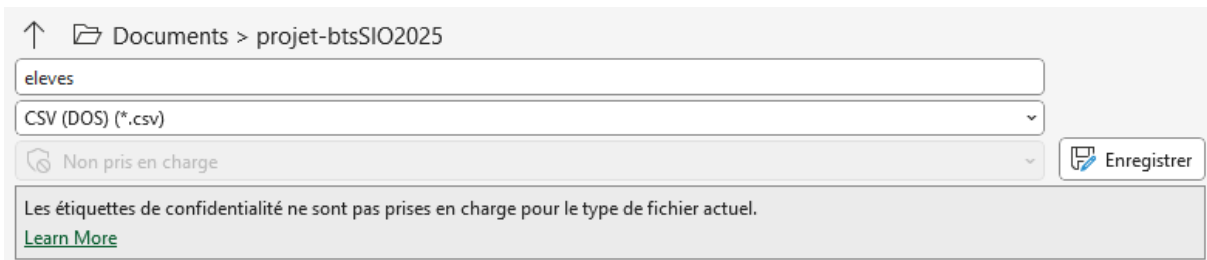
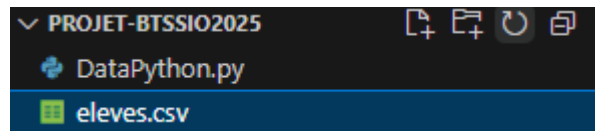
	Nom	Modifié le	Type	Taille
	DataPython	05/06/2025 11:54	Fichier source Pyt...	5 Ko
	eleves	03/06/2025 16:34	Fichier CSV Micro...	3 Ko
	Rapport	05/06/2025 15:31	Document Micros...	629 Ko

### 1 – Lecture des données :

En premier lieu je crée et j'ouvre un fichier Excel qu'on va convertir en CSV pour que python puisse le lire (Veiller à ce que le fichier figure dans le dossier **projetbtsSIO2025**, une fois converti) :

(Extrait du tableau Excel)

	A	B	C	D	E	F	G
	Prenom	Nom	Telephone	Adresse	Email	Absences	Moyenne
1	Lucas	Moreau	06 17 82 45 23	12 rue des Lilas, 91000 Evry	<a href="mailto:lucas.moreau@bts-sio.fr">lucas.moreau@bts-sio.fr</a>	3	13.75
2	Clara	Dubois	06 12 34 56 78	45 av. Victor Hugo, 91700 Ste-Geneviève	<a href="mailto:clara.dubois@bts-sio.fr">clara.dubois@bts-sio.fr</a>	1	15.20
3	Nathan	Lefèvre	06 22 89 45 12	8 allée des Chênes, 91300 Massy	<a href="mailto:nathan.lefevre@bts-sio.fr">nathan.lefevre@bts-sio.fr</a>	5	11.60
4	Emma	Laurent	06 11 67 89 02	3 rue Pasteur, 91400 Orsay	<a href="mailto:emma.laurent@bts-sio.fr">emma.laurent@bts-sio.fr</a>	2	16.10
5	Hugo	Bernard	06 93 45 78 90	27 rue du Parc, 91700 Viry-Châtillon	<a href="mailto:hugo.bernard@bts-sio.fr">hugo.bernard@bts-sio.fr</a>	6	10.40
6				89 bd			

*(Conversion en CSV)*

*(Résultat final)*

Et je procède à l'installation de **pandas** (manipulation de données) sur VSCODE, pour ce faire via le terminal :

➔ Entrez la commande suivante :

```
PS C:\Users\Frédéric BITSINDOU.LAPDXMHFW2\Documents\projet-btsSIO2025> pip install pandas
```

Enfin, « **import pandas as pd** » j'importe la bibliothèque **pandas** en attribuant le raccourci **pd** puis pour pouvoir lire le fichier CSV dans DataPhyton.py ➔ « **df = pd.read\_csv('eleves.csv', encoding='latin1', sep=';')** » (eleves.csv est le nom du fichier, encoding='latin1' permet de lire les caractères spéciaux en l'occurrence les accents, sep=';' car les colonnes sont séparées par des points-virgules, **df** est un nom de variable qui comporte le tableau en question) :



```
DataPython.py > ...
1 import pandas as pd
2 '''Etape 1 : Lectures des données'''
3 # Charger le fichier CSV
4 df = pd.read_csv('eleves.csv', encoding='latin1', sep=';')
5
6 # Afficher les 5 premières lignes
7 print(df.head())
8
```

Ci-dessous, un aperçu via le terminal de la commande « **python DataPython.py** » permettant d'exécuter et afficher le résultat obtenu à la suite du code précédemment effectué :

	Prenom	Nom	Telephone	Adresse	Email	Absences	Moyenne
0	Lucas	Moreau	06 17 82 45 23	12 rue des Lilas, 91000 Evry	lucas.moreau@bts-sio.fr	3	13.75
1	Clara	Dubois	06 12 34 56 78	45 av. Victor Hugo, 91700 Ste-Genevieve	clara.dubois@bts-sio.fr	1	15.20
2	Nathan	Lefvre	06 22 89 45 12	8 allée des Chaines, 91300 Massy	nathan.lefevre@bts-sio.fr	5	11.60
3	Emma	Laurent	06 11 67 89 02	3 rue Pasteur, 91400 Orsay	emma.laurent@bts-sio.fr	2	16.10
4	Hugo	Bernard	06 93 45 78 90	27 rue du Parc, 91700 Viry-Chitillon	hugo.bernard@bts-sio.fr	6	10.40

## 2 – Analyse des données :

Je procède à l'installation de **matplotlib** (visualisation de données) via le terminal grâce à la commande suivante :

```
PS C:\Users\Frédéric BITSINDOU.LAPDXMHFW2\Documents\projet-btsSIO2025> pip install matplotlib
```

Je fais une analyse globale des données en les explorant pour voir s'il y a des tendances intéressantes en l'occurrence la moyenne, absences, cohérence, etc.. :

- ➔ Afficher les informations générales via le code suivant (**df** : tableau contenant les informations des élèves) :

```
'''Afficher les informations générales'''
print("\n--- Informations générales sur les données ---")
print(df.info())
print("\n--- Statistiques descriptives ---")
print(df.describe())
```

- ➔ Vérifier s'il manque des données dans certaines colonnes (valeur vide ou « null ») :

```
'''Vérifier s'il y a des valeurs manquantes'''
print("\n--- Valeurs manquantes ---")
print(df.isnull().sum())
```

- ➔ Afficher les moyennes des élèves ainsi que les élèves ayant cumulé le plus d'absences :

```
'''Afficher les moyennes les plus hautes/basses'''
print("\n--- Élèves avec la meilleure moyenne ---")
print(df.sort_values(by="Moyenne", ascending=False).head(5))

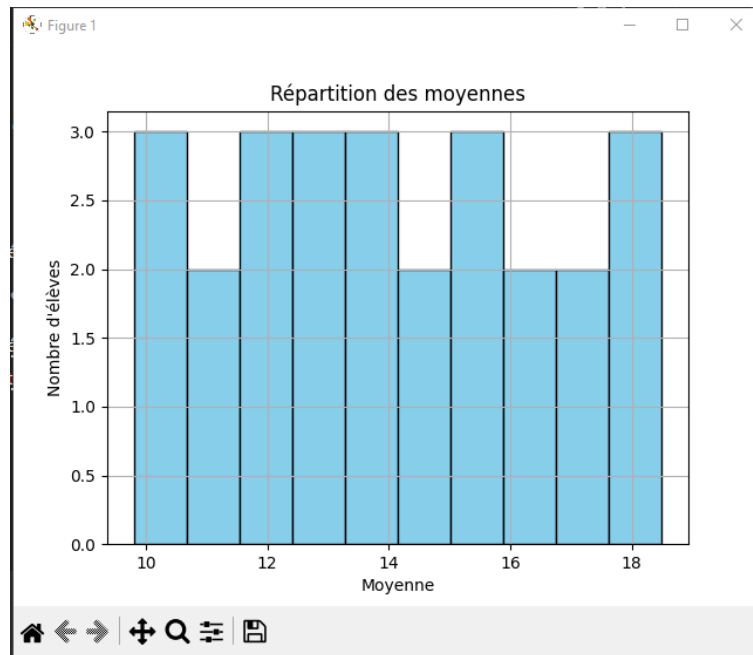
print("\n--- Élèves avec le plus d'absences ---")
print(df.sort_values(by="Absences", ascending=False).head(5))
```

- ➔ Ces paramètres via le code suivant confèrent un histogramme des moyennes (**plt.title()** : titre du graphique, **plt.xlabel()** : nom de l'axe horizontal, **plt.ylabel()** : nom de l'axe vertical, **grid(True)** : ajoute pour une lecture sans encombre et **plt.show()** : affiche le graphique :

```
import matplotlib.pyplot as plt

# Histogramme des moyennes
plt.hist(df['Moyenne'], bins=10, color='skyblue', edgecolor='black')
plt.title("Répartition des moyennes")
plt.xlabel("Moyenne")
plt.ylabel("Nombre d'élèves")
plt.grid(True)
plt.show()
```

(Résultats du graphique)



(Résultats obtenus via le terminal grâce aux précédents codes)

```

--- Informations générales sur les données ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Prenom      26 non-null    object
1   Nom         26 non-null    object
2   Telephone   26 non-null    object
3   Adresse     26 non-null    object
4   Email       26 non-null    object
5   Absences    26 non-null    int64
6   Moyenne     26 non-null    float64
dtypes: float64(1), int64(1), object(5)
memory usage: 1.6+ KB
None

--- Statistiques descriptives ---
      Absences  Moyenne
count  26.000000  26.000000
mean    3.038462  14.067308
std     2.615045   2.670728
min     0.000000   9.800000
25%     1.000000  11.887500
50%     2.500000  13.850000
75%     4.750000  16.025000
max     9.000000  18.500000

--- Valeurs manquantes ---
Prenom      0
Nom         0
Telephone   0
Adresse     0
Email       0
Absences    0
Moyenne     0
dtype: int64

```

--- élèves avec la meilleure moyenne ---									
	Prenom	Nom	Telephone		Adresse	Email	Absences	Moyenne	
25	Romane	Reynaud	06 70 60 50 40	4	rue du Pressoir, 91190 Gif-sur-Yvette	romane.reynaud@bts-sio.fr	0	18.5	
11	Manon	Nicolas	06 88 77 99 00	30	rue Gambetta, 91130 Ris-Orangis	manon.nicolas@bts-sio.fr	0	18.1	
5	Lea	Petit	06 54 76 89 01	89	boulevard Voltaire, 91120 Palaiseau	lea.petit@bts-sio.fr	0	17.9	
21	Lola	Masson	06 88 11 22 33	10	rue de l'Avenir, 91150 Morigny	lola.masson@bts-sio.fr	0	17.6	
15	Ines	Caron	06 84 56 23 19	29	rue des Tilleuls, 91150 Etampes	ines.caron@bts-sio.fr	0	17.4	
--- élèves avec le plus d'absences ---									
	Prenom	Nom	Telephone	...		Email	Absences	Moyenne	
8	Enzo	Roux	06 31 55 76 84	...	enzo.roux@bts-sio.fr	9	9.80		
10	Arthur	Colin	06 76 54 32 10	...	arthur.colin@bts-sio.fr	8	10.90		
16	Adam	Gauthier	06 39 12 88 45	...	adam.gauthier@bts-sio.fr	7	10.10		
18	Leo	Perrin	06 45 66 77 88	...	leo.perrin@bts-sio.fr	6	11.75		
4	Hugo	Bernard	06 93 45 78 90	...	hugo.bernard@bts-sio.fr	6	10.40		

### 3 – Détermination d'une loi simple :

Je devais au préalable établir la sous étape-préparatoire de la géolocalisation de l'adresse de chaque élève, pour ce faire :

➔ Procéder à l'installation de l'outil **geopy** (calcul de distances géographiques) via le terminal :

```
PS C:\Users\Frédéric BITSINDOU.LAPDXMHFW2\Documents\projet-btsSIO2025> pip install geopy
```

Ensuite j'importe les bibliothèques nécessaires (**pandas** : lire et manipuler fichier CSV, **geopy** : géolocaliser et calculer les distances, **time** : pour insérer des pauses entre les requêtes) :

```
'''sous-étape préparatoire qui précède la détermination d'une loi simple'''
import pandas as pd
from geopy.geocoders import Nominatim
from geopy.distance import geodesic
import time
```

J'initialise geolocator pour interagir avec le service Nominatim (géolocalisation gratuite basée sur OpenStreetMap) :

```
# Initialiser le géocodeur
geolocator = Nominatim(user_agent="bts_sio_stage")
```

Je vais géolocaliser l'adresse du lycée pour obtenir ses coordonnées GPS qui serviront ensuite de référence pour calculer la distance domicile-école de chaque élève :

```
# Adresse du lycée
adresse_lycee = "Rue Freteau de Peny, 77000 Melun"
coord_lycee = geolocator.geocode(adresse_lycee)
coord_lycee = (coord_lycee.latitude, coord_lycee.longitude)
```

Je procède à la création de la fonction `get_distance` pour tenter de géolocaliser l'adresse de l'élève qui prend en paramètre l'adresse d'un élève (`adresse_eleve`) afin de déterminer par un calcul la distance domicile-école et l'essai (`essai=3`) ; en accorde 3 en cas d'erreurs ou de timeout et retourne `None` si aucun résultat n'a été trouvé :

```
# Calculer la distance domicile → lycée
def get_distance(adresse_eleve, essais=3):
    for tentative in range(essais):
        try:
            location = geolocator.geocode(adresse_eleve, timeout=10)
            if location:
                coord_eleve = (location.latitude, location.longitude)
                return round(geodesic(coord_eleve, coord_lycee).km, 2)
            else:
                print(f"XX Adresse non trouvée : {adresse_eleve}")
                return None
        except Exception as e:
            print(f"Tentative {tentative+1}/{essais} échouée pour : {adresse_eleve}")
            print("Raison :", e)
            time.sleep(2) # attendre avant de réessayer
    print(f"XX Abandon de l'adresse après {essais} tentatives : {adresse_eleve}")
    return None
```

J'utilise une boucle for, pour chaque élève, je complète l'adresse avec « France » (important pour aider la géolocalisation), j'affiche l'adresse traitée grâce au print, on appelle la fonction `get_distance` pour calculer la distance et j'ajoute la distance dans une liste (`Distances = []` – l'initialisation) :

```
# Appliquer à chaque élève
distances = []
for adresse in df['Adresse']:
    adresse = adresse + ", France"
    print(f"Géocodage : {adresse}")
    distance = get_distance(adresse)
    distances.append(distance)
    time.sleep(1) # pour ne pas bloquer le service
```

J'ajoute les distances obtenues dans une nouvelle colonne du tableau « `df` » puis j'affiche par un print les 5 premières lignes pour vérifier que tout fonctionne bien :

```
# Ajouter la colonne "Distance"
df['Distance'] = distances

# Afficher un aperçu
print(df[['Nom', 'Adresse', 'Distance']].head())
```

(Résultats finaux)

```
Géocodage : 8 allée des Chenes, 91300 Massy, France
XX Adresse non trouvée : 8 allée des Chenes, 91300 Massy, France
Géocodage : 3 rue Pasteur, 91400 Orsay, France
XX Adresse non trouvée : 3 rue Pasteur, 91400 Orsay, France
Géocodage : 27 rue du Parc, 91700 Viry-Chatillon, France
Géocodage : 89 boulevard Voltaire, 91120 Palaiseau, France
Géocodage : 17 rue Jean Moulin, 91100 Corbeil-Essonnes, France
XX Adresse non trouvée : 17 rue Jean Moulin, 91100 Corbeil-Essonnes, France
Géocodage : 6 rue du Château, 91290 Arpajon, France
Géocodage : 22 rue des Peupliers, 91600 Savigny-sur-Orge, France
```

	Nom	Adresse	Distance
0	Moreau	12 rue des Lilas, 91000 Evry	14.97
1	Dubois	45 avenue Victor Hugo, 91700 Sainte-Genevieve-...	27.59
2	Lefevre	8 allée des Chenes, 91300 Massy	NaN
3	Laurent	3 rue Pasteur, 91400 Orsay	NaN
4	Bernard	27 rue du Parc, 91700 Viry-Chatillon	25.58

Ce que représente réellement la Distance (ex :14.97) dans ce tableau. C'est la distance géographique réelle (en kilomètres) entre l'adresse de l'élève et le lycée Léonard de Vinci à Melun.

Exemple : « 12 rue des Lilas, 91000 Evry » est transformée en coordonnées GPS :

- ➔ Latitude : 48.6347
- ➔ Longitude : 2.4382

Le lycée est géolocalisé à quelque chose comme :

- ➔ Latitude : 48.5391
- ➔ Longitude : 2.6609

Et le programme suivant permet de calculer la distance en ligne droite entre les deux points, en kilomètre, comme lorsque tu mesures avec une règle sur une carte :

```
return round(geodesic(coord_eleve, coord_lycee).km, 2)
```

Ainsi, on peut en déduire que Lucas Moreau, qui habite à Evry, est à 14.97 km à vol d'oiseau du lycée à Melun.

Passons à la régression linéaire pour déterminer une loi, pour ce faire :

- ➔ Procéder à l'installation **matplotlib** (visualisation) & **scikit-learn** (régression linéaire) via le terminal :

```
PS C:\Users\Frédéric BITSINDOU.LAPDX\MHFW2\Documents\projet-btsSIO2025> pip install matplotlib scikit-learn
```

J'importe les bibliothèques nécessaires telles que **matplotlib** : pour les graphiques, **LinearRegression** & **scikit-learn** pour la régression linéaire puis **numpy** : pour les calculs numériques si besoin :

```
'''Détermination d'une loi simple par une régression linéaire'''
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np
```

Programme permettant la suppression de lignes où la distance ou la moyenne est manquante, pour ne garder que les données exploitables :

```
# Supprimer les lignes où Distance ou Moyenne est manquante
df_clean = df.dropna(subset=['Distance', 'Moyenne'])
```

Préparation des données d'entrée :

- **x** correspond à la distance domicile-école
- **y** correspond à la moyenne générale de l'élève

```
# Variables X (distance) et y (moyenne)
X = df_clean[['Distance']] # X doit être une colonne en format tableau 2D
y = df_clean['Moyenne']    # y peut rester une série
```

Création du modèle de régression linéaire :

- Ce dernier va chercher la meilleure droite (loi) qui approxime la relation entre distance et moyenne :

```
# Création du modèle
modele = LinearRegression()
modele.fit(X, y)
```

Extraction des coefficients de la droite  $y = a * x + b$  :

- **a** : pente (impact de la distance sur la moyenne)
- **b** : valeur de la moyenne si la distance = 0

```
# Récupération des paramètres de la loi
a = modele.coef_[0]
b = modele.intercept_

print(f"Loi déterminée : Moyenne = {a:.2f} * Distance + {b:.2f}")
```

(Résultat final)

```
Loi déterminée : Moyenne = -0.00 * Distance + 15.06
```



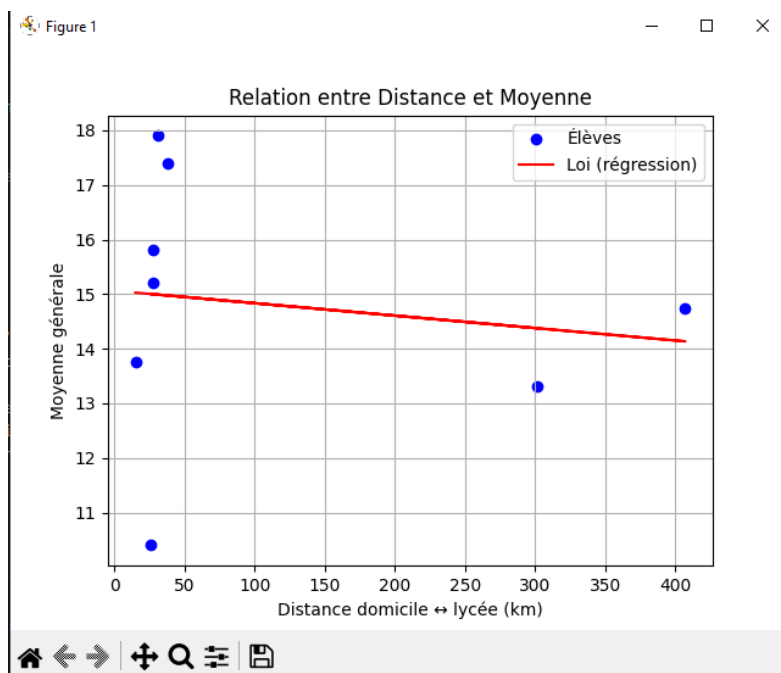
Je crée un graphique explicatif du code correspondant à ce dernier :

- ➔ Nuage de point bleu = élèves
- ➔ Droite rouge = tendance générale (loi simple)
- ➔ Affichage claire avec titre et étiquettes

```
# Affichage graphique
plt.scatter(X, y, color='blue', label='Élèves')
plt.plot(X, modele.predict(X), color='red', label='Loi (régression)')

plt.title("Relation entre Distance et Moyenne")
plt.xlabel("Distance domicile ↔ lycée (km)")
plt.ylabel("Moyenne générale")
plt.legend()
plt.grid(True)
plt.show()
```

(Résultat final)



Une fois que j'exécute le script ci-dessus on apercevra :

- ➔ Une droite rouge qui représente la loi trouvée
- ➔ La formule  $y = a * x + b$

Exemple : Loi déterminée : Moyenne =  $-0.25 * \text{Distance} + 16.20$ . Ainsi cela veut dire que plus un élève habite loin, plus sa moyenne est susceptible de diminuer de 0.25 points par km.

#### 4 – Prédiction à partir de la loi simple :

A la suite du script de l'étape précédente, je me sers du modèle de régression linéaire (à l'étape qui précède celle-ci) pour prédire la moyenne de chaque élève en fonction de sa distance au lycée. Les résultats seront ajoutés dans une nouvelle colonne appelée « Prédiction », voici le programme en question ci-dessous :

```
# Faire la prédiction avec la loi trouvée
df_clean['Prediction'] = modele.predict(X)
```

Cette partie affiche un tableau comparatif montrant :

- ➔ Le nom de l'élève
- ➔ La distance entre son domicile et le lycée
- ➔ Sa moyenne réelle
- ➔ Sa moyenne prédite par la formule trouvée

Cela permet de mesurer la fiabilité de la prédiction, en comparant la théorie à la réalité

```
# Afficher quelques lignes comparatives
print("\n--- Comparaison Réalité vs Prédiction ---")
print(df_clean[['Nom', 'Distance', 'Moyenne', 'Prediction']].head())
```

Ici, je simule un cas fictif : on veut savoir quelle serait la moyenne estimée d'un élève qui habite à 20 km du lycée. Le programme utilise la formule mathématique générée pour calculer cette moyenne automatiquement, ci-dessous :

```
distance_test = 20 # km
moyenne_predite = modele.predict([[distance_test]])
print(f"\nUn élève habitant à {distance_test} km aurait une moyenne estimée de : {moyenne_predite[0]:.2f}")
```

(Résultat final via le terminal)

```
--- Comparaison Réalité vs Prédiction ---
   Nom  Distance  Moyenne  Prediction
0  Moreau    14.97    13.75    15.026953
1  Dubois    27.59    15.20    14.998225
4  Bernard    25.58    10.40    15.002801
5   Petit    31.06    17.90    14.990326
7  Garcia   407.12    14.75    14.134254
```

```
Un élève habitant à 20 km aurait une moyenne estimée de : 15.02
PS C:\Users\Frédéric BITSINDOU.LAPDXMHFW2\Documents\projet-btsSIO2025> █
```

## 5 – Analyse des résultats :

Je vais évaluer pour voir si la loi mathématique trouvée est fiable en comparant :

- ➔ La moyenne réelle de chaque élève
- ➔ La moyenne prédite par la régression

En analysant l'écart (erreur) entre les deux.

En premier lieu je crée une nouvelle colonne appelée « Ecart » qui contient la différence entre la moyenne réelle de chaque élève (Moyenne) et la moyenne prédite par le modèle (Prédiction). Un écart proche de 0 indique que la prédiction est proche de la réalité :

```
# Calcul de l'écart entre la moyenne réelle et prédite
df_clean['Ecart'] = df_clean['Moyenne'] - df_clean['Prediction']
```

Ensuite j'affiche les 10 premières lignes du tableau final pour observer :

- ➔ Le nom de l'élève
- ➔ La distance par rapport au lycée
- ➔ Sa moyenne réelle
- ➔ La moyenne prédite
- ➔ L'écart entre les deux

Cela permet de voir concrètement si le modèle est cohérent ou non, grâce au code ci-dessous :

```
# Affichage des 10 premiers résultats
print("\n--- Écarts entre réalité et prédiction ---")
print(df_clean[['Nom', 'Distance', 'Moyenne', 'Prediction', 'Ecart']].head(10))
```

Je crée une ligne qui calcule l'erreur moyenne absolue :

- ➔ **abs(...)** prend la valeur absolue de chaque écart (pour ignorer les signes négatifs)
- ➔ **.mean()** calcule la moyenne des écarts

Ce chiffre représente une moyenne d'erreur globale du modèle, en point de moyenne.

```
# Calcul de l'erreur moyenne absolue
erreur_moyenne = abs(df_clean['Ecart']).mean()
```

Enfin j'affiche le résultat final du calcul précédent, arrondi à deux décimales. Le chiffre qu'on va obtenir permet d'évaluer la qualité générale de la prédiction :

```
print(f"\nErreur moyenne absolue : {erreur_moyenne:.2f} points")
```

« **Erreur moyenne absolue** » signifie qu'en moyenne, les prédictions se trompent d'environ 1.74 points, ce qui peut s'avérer décent ou élevé, selon le contexte.

*(Résultat final via le terminal)*

```
--- Ecart entre réalité et prédiction ---
      Nom Distance Moyenne Prediction   Ecart
0   Moreau   14.97   13.75   15.026953 -1.276953
1   Dubois   27.59   15.20   14.998225  0.201775
4   Bernard   25.58   10.40   15.002801 -4.602801
5     Petit   31.06   17.90   14.990326  2.909674
5     Petit   31.06   17.90   14.990326  2.909674
7   Garcia  407.12   14.75   14.134254  0.615746
9  Marchand   27.86   15.80   14.997610  0.802390
15   Caron   37.87   17.40   14.974823  2.425177
20   Blanc  301.36   13.30   14.375008 -1.075008

Erreur moyenne absolue : 1.74 points
```

## SYNTHESE PROJET D'ANALYSE PYTHON – BTS SIO

### Lecture des données :

J'ai commencé par importer un fichier CSV contenant :

- les noms et prénoms des élèves,
- leurs adresses postales,
- leurs absences et leurs moyennes.

J'ai utilisé la bibliothèque pandas pour charger et manipuler ces données efficacement.

### Analyse des données :

J'ai vérifié que les données étaient cohérentes et j'ai affiché des statistiques de base (moyenne, min, max).

J'ai aussi créé un histogramme de la répartition des moyennes avec matplotlib, pour visualiser la dispersion des résultats scolaires.

Sous-étape : calcul de la distance

À l'aide des bibliothèques geopy et geodesic, j'ai transformé les adresses des élèves en coordonnées GPS, puis j'ai calculé la distance à vol d'oiseau entre chaque élève et le lycée.

Des vérifications ont été faites pour gérer les erreurs de géocodage (ex : adresses non reconnues). Cela m'a appris à gérer des anomalies en autonomie.

### **Détermination d'une loi simple :**

J'ai ensuite cherché une relation mathématique entre la distance et la moyenne grâce à une régression linéaire.

Cela m'a permis d'obtenir une formule du type :

$$\text{Moyenne} = a * \text{Distance} + b$$

J'ai affiché cette loi sur un graphique représentant la tendance générale.

### **Prédiction et analyse des résultats :**

J'ai utilisé cette loi pour :

- prédire la moyenne d'un élève en fonction de la distance,
- comparer la moyenne réelle et la moyenne prédite,
- analyser l'écart entre les deux.

J'ai mesuré l'erreur moyenne absolue, ce qui m'a permis de juger de la fiabilité du modèle.

### **Conclusion :**

Ce projet m'a permis de mobiliser :

- des compétences techniques (Python, bibliothèques spécialisées),
- des compétences méthodologiques (structuration d'un projet),
- et des compétences analytiques (visualisation, interprétation).

J'ai appris à faire face aux limites des outils (géocodage, nettoyage de données) et à adapter mon raisonnement à la réalité.

Ce projet est une démonstration concrète de l'usage de Python pour résoudre un problème métier basé sur des données réelles.

Résumé du projet : Analyse de données avec Python

Étapes : lecture CSV, nettoyage, géolocalisation, régression linéaire, prédiction

Réalisé par Frédéric BITSINDOU - BTS SIO 2025